Fine-Grained, Multi-Domain Network Resource Abstraction as a Fundamental Primitive to Enable High-Performance, Collaborative Data Sciences

Qiao Xiang$^{\flat\ddagger}$, J. Jensen Zhang$^{\flat}$, X. Tony Wang$^{\flat}$, Y. Jace Liu$^{\flat}$,
Chin Guok†, Franck Le$^{\diamond}$, John MacAuley†, Harvey Newman*, Y. Richard Yang$^{\flat\ddagger}$,
$^{\flat}$Tongji University, ‡Yale University, †Lawrence Berkeley National Laboratory,
$^{\diamond}$IBM T.J. Watson Research Center, *California Institute of Technology,
{qiao.xiang, yry}@cs.yale.edu, {jingxuan.zhang, 13xinwang}@tongji.edu.cn,
yang.jace.liu@linux.com, {chin, macauley}@es.net, fle@us.ibm.com, newman@hep.caltech.edu

***Abstract*—Multi-domain network resource reservation systems are being deployed, driven by the demand and substantial benefits of providing predictable network resources. However, a major lack of existing systems is their coarse granularity, due to the participating networks' concern of revealing sensitive information, which can result in substantial inefficiencies. This paper presents Mercator, a novel multi-domain network resource discovery system to provide fine-grained, global network resource information, for collaborative sciences. The foundation of Mercator is a resource abstraction through algebraic-expression enumeration (*i.e.*, linear inequalities/equations), as a compact representation of the available bandwidth in multi-domain networks. In addition, we develop an obfuscating protocol, to address the privacy concerns by ensuring that no participant can associate the algebraic expressions with the corresponding member networks. We also introduce a super-set projection technique to increase Mercator's scalability. Finally, we implement Mercator and demonstrate both its efficiency and efficacy through extensive experiments using real topologies and traces.**

***Index Terms*—Multi-domain networks, resource discovery, privacy preserving**

I. Introduction

Many of today's premier science experiments, such as the Large Hadron Collider (LHC) [1], the Square Kilometre Array (SKA) [2], and the Linac Coherent Light Source (LCLS) [3], rely on finely-tuned workflows that coordinate geographically distributed resources (*e.g.*, instrument, compute, storage) to enable scientific discoveries. An example of this is the movement of LHC data from Tier 0 (*i.e.*, the data center at European Organization for Nuclear Research, known as CERN) to Tier 1 (*i.e.*, national laboratories) storage sites around the world. This requires deadline scheduling to keep up with the amount of information that is continually generated by instruments when they are online. Another example is the "superfacility" model being developed by LCLS to allow streaming of data from instruments, across the Wide-Area Network (WAN), directly into supercomputers' burst buffers for near real-time analysis. The key to supporting these distributed resource workflows is the ability to reserve and guarantee bandwidth across multiple network domains to facilitate predictable end-to-end network connectivity. As such, several

The corresponding authors are Qiao Xiang and Y. Richard Yang.

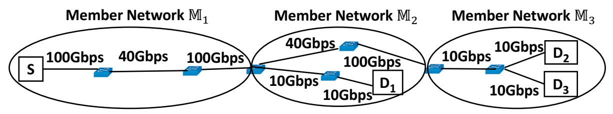

Member Network $\mathbb{M}_1$
Member Network $\mathbb{M}_2$
Member Network $\mathbb{M}_3$
S
100Gbps
40Gbps
100Gbps
40Gbps
100Gbps
10Gbps
10Gbps
D1
10Gbps
10Gbps
D2
10Gbps
D3

Fig. 1: A motivating example where a user wants to reserve bandwidth for three source-destination pairs: (S, D_1), (S, D_2) and (S, D_3), across 3 member networks $\mathbb{M}_1$, $\mathbb{M}_2$ and $\mathbb{M}_3$.

Research and Education (R&E) networks have deployed inter-domain circuit reservation systems. For example, the Energy Sciences Network (ESnet), a network supporting the LHC experiments, has deployed an On-Demand Secure Circuits and Advance Reservation System called OSCARS [4].

However, due to networks' concern of revealing sensitive information, existing systems do not provide a network interface for users to access network resource information (*e.g.*, network capabilities). Instead, they only allow users to submit requests for reserving a specific amount of bandwidth, and return either success or failure [4]–[10]. This approach, which we call "probe requests" in the rest of this paper, often results in poor performance and fairness. Specifically, while solutions for reserving bandwidth within a single member network, can be very efficient, solutions for discovering and reserving bandwidth for correlated and concurrent flows across multiple member networks face unique challenges. In particular, solutions to reserving bandwidth within a single member network are often provided with the member network's topology, and links' availability. In contrast, because this information is typically considered sensitive, member networks do not reveal internal network details to external parties. As a result, existing multi-domain reservation systems treat each member network as a black box, probe their available resource by submitting varied circuit reservation requests, and receive boolean responses. In other words, current solutions perform a depth-first search on all member networks, and rely on a trial and error approach: to reserve bandwidth, repeated, and varied attempts may have to be submitted until success.

To illustrate the limitations of existing systems, we consider a collaboration network composed of three member networks running OSCARS [4], as shown in Fig. 1. A user may submit a request to reserve bandwidth for three circuits, from source host S to destination hosts D_1, D_2 and D_3. Given the capabilities of the source host (*e.g.*, the source host may have a 100 Gbps network card), and to ensure fairness across the circuits, the user may request 33.33 Gbps for each circuit. Upon receiving this request, OSCARS processes the circuits sequentially, for example, in the order of (S, D_1), (S, D_2) and (S, D_3). For each circuit, it uses a depth-first search approach to probe if each member network can provide the requested bandwidth. In this example, there is no path with 33.33 Gbps of bandwidth from S to D_1, and hence OSCARS notifies the user that this request fails.

The user can then adjust the requested bandwidth. However, with the limited feedback in OSCARS, the user does not know the amount of available bandwidth from S to D_1. Consequently, the user may use a cut-to-half-until-reserved search strategy. As a result, after 12 attempts, the networks allocate 8.33 Gbps ($33.33 \rightarrow 16.67 \rightarrow 8.33$) for (S, D_1), 8.33 Gbps ($33.33 \rightarrow 16.67 \rightarrow 8.33$) for (S, D_2) and 1.04 Gbps ($33.33 \rightarrow 16.67 \rightarrow 8.33 \rightarrow 4.17 \rightarrow 2.08 \rightarrow 1.04$) for (S, D_3). In addition to requiring a large number of search attempts, the approach may obtain a bandwidth allocation that is far from optimal. For example, given the links' capacities and availability, a fair optimal bandwidth allocation is actually 5 Gbps for each circuit. Without a network interface to provide network resource information, designing an algorithm using existing systems to identify this solution can lead to substantially more complexity and churns.

In addition to multi-domain circuit reservation systems, multiple multi-domain resource discovery systems have been developed and deployed (*e.g.*, [11]–[17]). However, these systems focus on the discovery of endpoint resources (*i.e.*, computation and storage resources) and their availability for different services. They do not provide a network interface for applications to discover the network resource availability and sharing properties [18]–[20].

In this paper, we present Mercator, a novel multi-domain resource discovery system designed to optimize large, multi-domain transfers, and address the limitations of current reservation systems through three main components. The first and core component of Mercator is a resource abstraction through algebraic-expression enumeration (*i.e.*, linear inequalities and equations), which provides a compact, unifying representation of multi-domain network available bandwidth. For example, considering the same example of Fig. 1, the resource abstraction captures the constraints from all networks using the set of linear inequalities depicted in Fig. 2. Specifically, the variables x_1, x_2, x_3 represent the available bandwidth that can be reserved for (S, D_1), (S, D_2) and (S, D_3), respectively. Each linear inequality represents a constraint on the reservable bandwidths over different shared resources by the three circuits. For example, the inequality $x_1 + x_2 + x_3 \leq 100$ indicates that all three circuits share a common resource and that the sum of their bandwidths can not exceed 100 Gbps. With this set of linear inequalities, the user does not need to repeatedly probe the domains, but can immediately derive the bandwidth allocation to satisfy its own objective (*e.g.*, same

$\mathbb{M}_1$:

$x_1 + x_2 + x_3 \leq 100,$
$x_1 + x_2 + x_3 \leq 40,$
$x_1 + x_2 + x_3 \leq 100,$

$\mathbb{M}_2$:

$x_2 + x_3 \leq 40, \quad x_1 \leq 10,$
$x_2 + x_3 \leq 100, \quad x_1 \leq 10,$

$\mathbb{M}_3$:

$x_2 + x_3 \leq 10,$
$x_2 \leq 10,$
$x_3 \leq 10,$

Fig. 2: Illustration of resource abstraction for the reservation request from Fig. 1.

rate for each transfer, different ratios according to demand ratios, or a fairness allocation such as max-min fairness).

Second, Mercator introduces a resource abstraction obfuscating protocol to ensure that member networks and other external parties cannot associate an algebraic expression with a corresponding member network, leading to a complete unified aggregation of multiple domains, appearing as much as possible as a single (virtual) network. Although such complete integration may not be needed in all settings, it can be highly beneficial in settings with higher privacy or security concerns. For example, in the scenario of Fig. 1, this protocol ensures that (1) the user cannot infer that the constraint $x_2 + x_3 \leq 10$ comes from network $\mathbb{M}_3$, and (2) that neither network $\mathbb{M}_1$ nor $\mathbb{M}_2$ knows the existence of this constraint. Finally, Mercator also introduces a super-set projection technique, which substantially improves the scalability and performance of Mercator through pre-computation and projection.

The main contributions of this paper are as follows:

- We identify the fundamental reason of the poor performance of current reservation systems for multi-domain data transfers as the lack of visibility of network topology and link availability of each member network, and design Mercator, a novel multi-domain network resource discovery system, to address this issue;
- In Mercator, we propose a novel, compact resource abstraction to represent the network resource availability and sharing, *e.g.*, bandwidth, among virtual circuit requests through algebraic-expression enumeration;
- We design a resource abstraction obfuscating protocol to prevent the user from associating the received algebraic expressions with their corresponding member networks;
- We develop a super-set projection technique to substantially improve the scalability of Mercator;
- We fully implement Mercator and conduct extensive experiments using real network topologies and traces. Results show that Mercator (1) efficiently discovers available networking resources in collaborative networks on average 2 orders of magnitude faster, and allows fairer allocations of network resources; (2) preserves the member networks' privacy with little overhead; and (3) scales to a collaborative network of 200 member networks.

The remaining of this paper is organized as follows. We give an overview of Mercator in Section II. We give the details of the algebraic-expression-based resource abstraction in Section III. We discuss the resource abstraction obfuscating protocol and the super-set projection technique in Section IV and Section V, respectively. We present the evaluation results of Mercator in Section VI. We discuss the related work in Section VII and conclude the paper in Section VIII.

II. Mercator Overview

This section presents the basic workflow and the architecture of Mercator, and a brief overview of its three main components: the resource abstraction through algebraic-expression enumeration, the resource abstraction obfuscating protocol and the super-set projection technique.

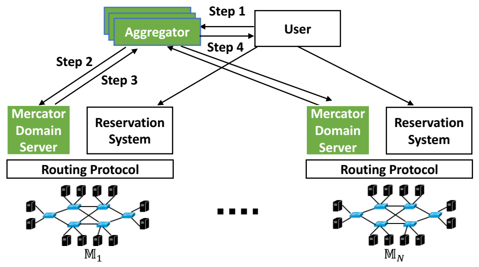


Fig. 3: The architecture and basic workflow of Mercator.

A. Basic Workflow

Mercator introduces and relies on a logically centralized aggregator, and a Mercator domain server in each member network. Consider a multi-domain network of N member networks $\mathbb{M}_i$, where $i = 1, \ldots, N$ (Fig. 3). The basic workflow of Mercator to discover the multi-domain network bandwidth availability and sharing for a set of requested circuits is:

- Step 1: A user (*e.g.*, an application) submits a resource discovery request for a set of circuits to the aggregator by specifying the source and destination endpoints of each circuit.
- Step 2: After authenticating and verifying the authorization of the request, the aggregator determines the member networks that the circuits traverse, and queries the Mercator domain server in each of these member networks to discover their resource abstractions. The determination of the relevant member networks for the aggregator to contact is further described in Section II-B.
- Step 3: Upon receiving the query from the aggregator, each Mercator domain server computes the resource abstraction (Section II-C, Section III) of the corresponding member network, and executes an obfuscating protocol (Section II-C, Section IV) to send the obfuscated resource abstraction to the aggregator.
- Step 4: The aggregator collects the obfuscated resource abstractions from the relevant member networks, and derives the original resource abstractions to present to the user. Based on the received information, the user determines the bandwidth allocation for each circuit, and sends a reservation request to the underlying reservation system.

The above workflow illustrates the main steps for a user to discover the available network bandwidth and properties for a set of circuits traversing multiple member networks. To further improve the scalability of Mercator, Section V introduces the super-set projection technique. It allows the aggregator to proactively discover the resource abstractions for a set of circuits between every pair of source and destination member networks, and project the pre-computed result to get the resource abstraction when receiving actual requests from users. The super-set projection technique can significantly reduce the delay, as well as number of messages, of resource discovery, and allows the aggregator to process multiple requests concurrently.

B. Architecture

This section describes the roles of the aggregator and Mercator domain servers in further details (Fig. 3).

Aggregator: The aggregator is the main interface of Mercator. It is responsible for authenticating and verifying the authorization of users' resource discovery requests (*e.g.*, through PKI [21])[1], querying Mercator domain servers in member networks to discover network resource information, and returning the collected abstractions to users.

The aggregator has connections to Mercator domain servers in all member networks. It also acts as a Border Gateway Protocol (BGP) [24] speaker, and has BGP sessions to all member networks. Consequently, given a request for a set of circuits F, the aggregator can infer the member-network path for each circuit, *i.e.*, the list of member networks a circuit will traverse, and the ingress points of the circuits to each member network[2] (as described in Step 1 of workflow). As such, for this request, the aggregator can also infer the set of circuits traversing and consuming resources in each $\mathbb{M}_i$, denoted as F_i. It can then queries the Mercator domain servers at each $\mathbb{M}_i$ by providing F_i and their ingress points to enter $\mathbb{M}_i$.

Mercator domain server: Given a Mercator domain server in member network $\mathbb{M}_i$, its primary role is to compute the resource abstraction of $\mathbb{M}_i$. To achieve it, Mercator follows the layering design principle to separate the routing protocol and the available network resources. In this way, given a set of circuits sent by the aggregator, their routes in $\mathbb{M}_i$ are computed and provided by the routing protocol in $\mathbb{M}_i$. The Mercator domain server in $\mathbb{M}_i$ takes these routes as inputs, and derives the available bandwidth and shared properties for the requested flows along those routes. After computing the abstraction, the Mercator domain server executes an obfuscating protocol to send the obfuscated resource abstraction to the aggregator, which addresses member networks' privacy concern.

C. Key Design Points

Having illustrated the high-level workflow of Mercator, we next give a brief overview on its key design points.

Resource abstraction through algebraic-expression enumeration (Section III): Mercator follows two important principles in human-computer interaction, *familiarity* and *uniformity*, to design a unifying abstraction that captures the

[1]Mercator may adopt different authentication/authorization systems, *e.g.*, OpenID [22] and SAML [23], depending on the specific requirements of different collaborative science programs. We leave the detailed investigation of this issue in Mercator as future work.

[2]In BGP glossary, such a path is also called an autonomous-system-path, or an AS-path, which is announced in BGP update messages along BGP sessions. The Route View Project [25] relies on a similar architecture with BGP speakers establishing sessions with hundreds of peering networks to collect BGP updates, and provides a real time monitoring infrastructure. In particular, we observe that the AS path for each destination prefix is currently already collected and made publicly available. As such, Mercator does not introduce additional privacy issues.

properties (*e.g.*, available bandwidth) of resources shared – within and between member networks – by a set of requested circuits. This novel, compact resource abstraction is the core component of Mercator, and relies on algebraic expressions (*i.e.*, linear inequalities / equations), a concept familiar to scientists and network engineers [26], to express the available bandwidth sharing for a set of requested circuits to be reserved.

Existing resource abstractions, including graph-based abstractions [27], [28] and the one-big-switch abstractions [29], [30], either fail to protect the private, sensitive information of each member network, or fail to capture the resource sharing between virtual circuit requests. In contrast, the resource abstraction of Mercator, expressed through algebraic-expression enumeration, naturally and accurately captures the available bandwidth of shared resources by a set of circuits without requiring member networks to reveal their network topology. Compared with the Boolean response of current resource reservation systems such as OSCARS, the user receives the complete bandwidth feasible region of the collaboration networks for the requested circuits represented through algebraic expressions. A point in that feasible region represents a feasible allocation of bandwidth for the different circuits in the request. In other words, the user can choose any point in the returned region as the bandwidth parameters for the circuits to be reserved, using his own resource allocation strategy (*e.g.*, max-min fairness [31]).

Resource abstraction obfuscating protocol (Section IV): The algebraic-expression-based abstraction provides a compact, unifying representation of the multi-domain network resource information. It does not require member networks to reveal their network topologies and link availabilities. However, it does expose the bandwidth feasible region of each member network (illustrated by the examples in Section I and Section III). Some member networks might prefer not to expose such information, as malicious parties may use it to identify links where to launch attacks (*e.g.*, DDoS). To address this issue, we develop a resource abstraction obfuscating protocol. More specifically, the protocol prevents the resource discovery aggregator from identifying the source of each received resource constraint. The key idea consists of having each Mercator domain server obfuscate its own set of linear inequalities as a set of linear equations through a private random matrix of its own and a couple of random matrices shared with few other Mercator domain servers from other member networks (*e.g.*, through a consensus protocol), and then sends the obfuscated set of linear equations back to the aggregator using symmetric-key encryption, *e.g.*, Advanced Encryption Standard (AES) [32]. We demonstrate that from the received obfuscated equations, the aggregator can retrieve the actual bandwidth feasible region for the circuits across member networks, but cannot associate any linear inequality with its corresponding member network. As a result, even if a malicious party obtains the bandwidth feasible region across member networks, launching attacks to all member networks is much harder than attacking a particular member network.

Super-set projection (Section V): To improve the scalability of Mercator, we introduce the super-set projection technique. The main idea consists of having the aggregator periodically query Mercator domain servers to discover the resource abstraction for a set of circuits between every pair of source and destination member networks. With these precomputed abstractions, when a user submits a resource discovery request, the aggregator does not need to query the Mercator domain servers to compute the abstraction for each received request. Instead, the aggregator performs a projection on the precomputed abstractions based on the source and destination member networks of each circuit in the actual user request, to get the abstraction for this request. For example, consider a network of 2 member networks $\mathbb{M}_1$ and $\mathbb{M}_2$. Using super-set projection, the aggregator queries the Mercator domain servers at both member networks for a set of 2 circuits, one from $\mathbb{M}_1$ to $\mathbb{M}_2$ and the other from $\mathbb{M}_2$ to $\mathbb{M}_1$, and gets a set of linear inequalities $\{x_{12} + x_{21} \leq 100,\ x_{12} \leq 50\}$. Suppose later a user submits a request for 1 circuit, with the source being an endpoint in $\mathbb{M}_2$ and the destination being an endpoint in $\mathbb{M}_1$, to the aggregator. The aggregator projects the precomputed set of linear inequalities by removing all variables that are not x_{21}, and returns the result $\{x_{21} \leq 100\}$ to the user.

Such projection is much more efficient than having Mercator domain servers compute the abstraction for each received circuit request. With this technique, when a user submits a resource discovery request to the aggregator, the aggregator does not need to query Mercator domain servers (Step 2 in Section II-A), and the Mercator domain servers do not need to compute and obfuscate the resource abstraction for the request (Step 3 in Section II-A). Only when the user fails to reserve the resource based on the projected abstraction will the aggregator query the Mercator domain servers to obtain an up-to-date abstraction for the user. As such, servers in the aggregator pool can process requests concurrently (*e.g.*, using optimistic concurrency control), significantly improving the scalability, fault-tolerance, and performance of Mercator.

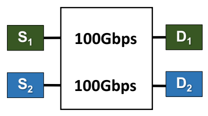
After an overview of the key design points in Mercator, we discuss these designs in detail in the next few sections.

III. Resource Abstraction through Algebraic-Expression Enumeration

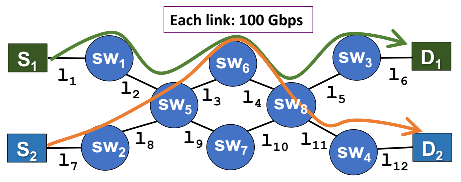
In this section, we give the details of the resource abstraction through algebraic-expression enumeration, the core component of Mercator. We first discuss the limitations of existing design options. Then we give the specifications of this abstraction. We also discuss how it handles important use cases, *e.g.*, multicast, multi-path routing and load balancing, in Appendix A.

Basic issue: As illustrated by the example in Section I, the fundamental reason for the poor performance of existing circuit reservation systems is they are lack of the visibility of properties, *e.g.*, bandwidth, of shared network resources for a set of circuits to be reserved. One may think of a strawman to let each member network provide the full topology information to the aggregator in a graph-based abstraction [27], [28]. This design, however, exposes all the sensitive, private information of each member network, *i.e.*, network topology and links' availability, to external parties, leading to security breaches.

A second strawman is to use a one-big-switch abstraction to provide simplified views of network information [29], [30], which protects the privacy of each member network. However, this abstraction fails to capture the information of shared resource among virtual circuit requests and thus is inaccurate. Consider the example in Fig. 4, where the user wants to reserve two circuits from S_1 to D_1 and S_2 to D_2, respectively. Using the one-big-switch abstraction in the P4P system [29], the user will get the information that each circuit can reserve a bandwidth up to 100 Gbps (Fig. 4a). However, the routes for the two circuits – computed by the underlying routing protocol – share common links l_3 and l_4 (Fig. 4b), making it infeasible for both circuits to each reserve a 100 Gbps bandwidth.



(a) The one-big-switch shows that each circuit can get a 100 Gbps bandwidth.



(b) The physical topology shows that the route of two circuits share bottleneck links, *i.e.*, l_3 and l_4, hence they can only collectively get a 100 Gbps bandwidth.

Fig. 4: A running example for illustrating the inefficiency of one-big-switch abstraction and the basic idea of resource abstraction through algebraic-expression enumeration, where two circuits (S_1, D_1) and (S_2, D_2) need to be reserved.

In some recent studies [33], [34], a variation of the one-big-switch abstraction was proposed to define the resource sharing among different traffic flows as operations defined in different algebra fields. However, this abstraction is too complex and can only handle single-path routing policies.

Basic idea: Different from the graph-based abstraction and the one-big-switch abstraction, the basic idea of the resource abstraction in Mercator is simple yet powerful: *given a set of requested circuits to be reserved, capture the properties (e.g., available bandwidth) of relevant shared resources, through a set of algebraic expressions.*

Specifically, suppose the Mercator domain server at a member network receives the resource discovery request of a set of circuits F entering this member network. For each circuit $f_j \in F$, we use x_j to denote the available bandwidth the user can reserve for this circuit. Upon receiving this request, the Mercator domain server first checks the intradomain route of each circuit f_j. Then the server enumerates all the links in the member network. For each link l_u, it generates a linear inequality:

$$\sum x_j \leq l_u.bandwidth, \forall f_j \text{ that uses link } l_u \text{ in its route.}$$

Revisit the example in Fig. 4, the Mercator domain server will generate the following set of linear inequalities $\Pi(F)$:

$$\begin{aligned} x_1 &\leq 100 && \forall l_u \in \{l_1, l_2, l_5, l_6\}, \\ x_2 &\leq 100 && \forall l_u \in \{l_7, l_8, l_{11}, l_{12}\}, \\ x_1 + x_2 &\leq 100 && \forall l_u \in \{l_3, l_4\}, \end{aligned} \tag{1}$$

which accurately captures the bandwidth sharing among two circuits' routes.

Removing redundant linear inequalities: Observe the set of linear inequalities in the above example. One may realize that this set has redundancies, *e.g.*, there are 4 same inequalities $x_1 \leq 100$ in this set. Given $\Pi(F)$, a linear inequality $y \in \Pi(F)$ is redundant if and only if the optimal solution of any optimization problem with $\Pi(F)$ as the constraint is the same as that with $\Pi(F) - \{c\}$ as the constraint. In our system, the Mercator domain server adopts a classic compression algorithm [35] to remove the redundant linear inequalities. In this example, the compressed $\Pi(F)$ will only contain one inequality, *i.e.*, $x_1 + x_2 \leq 100$.

Geometric interpretation of resource abstraction: Given $\Pi_i(F)$, the resource abstraction of $\mathbb{M}_i$ for a set of F circuits, from the geometric perspective, represents the bandwidth feasible region of $\mathbb{M}_i$ for providing bandwidths to this set of circuits. Therefore, given a set of F circuits spanning over N member networks, the union of $\Pi_i(F_i)$, where $F_i \subset F$ is the set of circuits that will consume resources in $\mathbb{M}_i$, represents the complete bandwidth feasible region of all N member networks for the requested circuits.

Through algebraic-expression enumeration, the resource abstraction can handle not only unicast, as shown above, but many other settings. In Appendix A, we show how it handles three important use cases in collaborative data sciences.

IV. Privacy-Preserving Resource Abstraction

Given a member network, the algebraic-expression-based resource abstraction accurately captures the shared available bandwidth among virtual circuits without exposing its network topology and links' availability. However, as shown in Section III, the geometric interpretation of a resource abstraction is that it represents the bandwidth feasible region of the corresponding member network for a set of circuits. Such information is still private and sensitive, and a malicious party who acquires it may use it to launch attacks to the corresponding member network. To preserve the privacy of bandwidth feasible region of member networks while still providing the accurate bandwidth sharing information for circuits, we develop a resource abstraction obfuscating protocol in Mercator. In this section, we first formally define the privacy-preserving resource abstraction problem. Next, we present the details of our protocol. We also conduct a rigorous analysis of our protocol in Appendix B.

A. Privacy-Preserving Resource Abstraction Problem

Basic issue: We use the example in Fig. 5 to illustrate the privacy concern of the resource abstraction, where Mercator tries to discover the shared bandwidth of two virtual circuits (S_1, D_1) and (S_2, D_2) across 3 member networks. In this example, all links in black line are 1 Tbps aggregating links. The inter-member-network-paths of two circuits are $[\mathbb{M}_1, \mathbb{M}_2]$ and $[\mathbb{M}_1, \mathbb{M}_3]$, respectively. And two circuits share the same intra-domain path in $\mathbb{M}_1$.

When receiving the resource discovery request, the Mercator domain server at each member network will abstract the bandwidth sharing of both circuits into a set of linear inequalities. After removing the redundant inequalities of each member

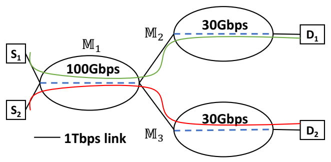


Fig. 5: A running example to illustrate the resource abstraction obfuscating.

network, the resource abstraction of each member network is:

$$\begin{aligned}&\Pi_1(F_1) : \{x_1 + x_2 \leq 100\}\\&\Pi_2(F_2) : \{x_1 \leq 30\}\\&\Pi_3(F_3) : \{x_2 \leq 30\}.\end{aligned} \tag{2}$$

If each Mercator domain server directly sends its own resource abstraction to the aggregator, the aggregator will have the knowledge of the bandwidth feasible region of each individual member network. This makes the whole collaboration network vulnerable because the aggregator is a single point of failure possessing the private information of all member networks. In other words, if an attacker gains the control to the aggregator, he can leverage such specific information to attack any member network.

Problem definition: To make Mercator functional and secure, therefore, we need a solution that provides the accurate bandwidth sharing information for the set of virtual circuits to be reserved, and at the same time protects each member network from exposing its private bandwidth feasible region. To this end, we first give a formal definition of privacy-preserving, equivalent resource abstraction:

***Definition** 1 (Equivalent, Privacy-Preserving Resource Abstraction):* Given a set of circuits F that span over $N > 1$ member networks, the resource abstraction $\Pi_p(F)$ collected by the aggregator is equivalent and privacy-preserving if (1) the bandwidth feasible region represented by $\Pi_p(F)$ is the same as that represented by $\cup_i \Pi(F_i)$ where $i = 1, 2, \ldots, N$; and (2) for any linear inequality $c \in \Pi_p(F)$, the aggregator cannot associate it with a particular member network.

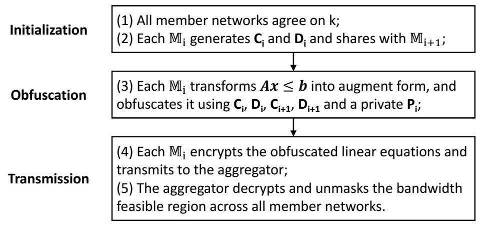
In this definition, $\Pi(F_i) \cup \Pi(F_j)$ means the union of two sets of linear inequalities. Geometrically speaking, it means the intersection of the feasible regions represented by $\Pi(F_i)$ and $\Pi(F_j)$. With this definition, we further define the privacy-preserving resource abstraction problem:

***Problem** 1 (Privacy-Preserving Resource Abstraction Problem):* Given a set of circuits F that span over $N > 1$ member networks, design a security protocol in the resource discovery system to ensure that (1) the aggregator receives the equivalent, privacy-preserving resource abstraction $\Pi_p(F)$; and (2) for any $\mathbb{M}_i$, it does not know any linear inequality from any other $\Pi_j(F_j)$, where $j \neq i$.

Security model: In this paper, we assume a *semi-honest* security model, *i.e.*, the aggregator and all member networks will not deviate from the security protocol, but merely try to gather information during the execution of the protocol [36]. This is sufficient for collaboration science networks where member networks share resources to collaboratively conduct common tasks such as data transfers, storage and analytics.

B. Resource Abstraction Obfuscating Protocol

There are different design options for Problem 1, *e.g.*, garbled circuit based protocols [37]. However, these designs would incur expensive computation and communication overhead, hence are not suitable for the need of multi-domain resource discovery. In this paper, we tackle this problem by designing a novel resource abstraction obfuscating protocol that only requires simple operations on matrices, *i.e.*, addition and multiplication.


Initialization

(1) All member networks agree on k;
(2) Each $\mathbb{M}_i$ generates $\mathbf{C_i}$ and $\mathbf{D_i}$ and shares with $\mathbb{M}_{i+1}$;

Obfuscation

(3) Each $\mathbb{M}_i$ transforms $\boldsymbol{Ax} \leq \boldsymbol{b}$ into augment form, and obfuscates it using $\mathbf{C_i}$, $\mathbf{D_i}$, $\mathbf{C_{i+1}}$, $\mathbf{D_{i+1}}$ and a private $\mathbf{P_i}$;

Transmission

(4) Each $\mathbb{M}_i$ encrypts the obfuscated linear equations and transmits to the aggregator;
(5) The aggregator decrypts and unmasks the bandwidth feasible region across all member networks.

Fig. 6: The resource abstraction obfuscating protocol.

Basic idea: Our protocol leverages random matrix theory [38], [39]. In particular, each $\mathbb{M}_i$ independently computes and sends to the aggregator a set of disguised *linear equations*, which are derived from the private $\Pi_i(F_i)$, a random matrix $\mathbf{P_i}$ known only to $\mathbb{M}_i$, two random matrices $\mathbf{C_i}$ and $\mathbf{D_i}$ known only to $\mathbb{M}_i$ and $\mathbb{M}_{i-1}$, and two random matrices $\mathbf{C_{i+1}}$ and $\mathbf{D_{i+1}}$ known only to $\mathbb{M}_i$ and $\mathbb{M}_{i+1}$.

Protocol: The protocol is composed of three phases: initialization, obfuscation and transmission, as shown in Fig. 6. For the simplicity of presentation, we let $m_i = |\Pi_i(F_i)|$, *i.e.*, the number of linear inequalities in $\Pi_i(F_i)$ after redundancy removal, and $M_i = \sum_{j=1}^{i} m_j$. During the *initialization phase*, all member networks agree on a common $k > \sum m_i$. For each $\mathbb{M}_i$ where $i = 1, 2, \ldots, N-1$, it generates a k-by-$(|F| + m_i + m_{i+1})$ random matrix $\mathbf{C_i} = [\mathbf{C_i^{|F|}}\ \mathbf{C_i}^{m_i}\ \mathbf{C_i}^{m_{i+1}}]$, and a k-by-1 random matrix $\mathbf{D_i}$, and sends to $\mathbb{M}_{i+1}$. And we define $\mathbf{C_0}$, $\mathbf{D_0}$, $\mathbf{C_N}$ and $\mathbf{D_N}$ as zero matrices. As we will illustrate in the remaining of this section, these zero matrices are used for presentation completeness and will not affect the correctness of the obfuscating protocol.

During the *obfuscation phase*, each $\mathbb{M}_i$ introduces m_i slack variables, denoted by $\mathbf{x_i^s}$, to transform $\Pi_i(F_i) = \mathbf{A_i x} \leq \mathbf{b_i}$ from the standard form to the augment form and gets the following equivalent linear system:

$$\begin{bmatrix}\mathbf{A_i} & \mathbf{I_{m_i}}\end{bmatrix}\begin{bmatrix}\mathbf{x}, & \mathbf{x_i^s}\end{bmatrix} = \mathbf{b_i}. \tag{3}$$

We then add slack variables introduced by all other member networks with zero coefficients into the linear system in Equation (3) and get the following equivalent linear system:

$$\begin{bmatrix}\mathbf{A_i} & \mathbf{0_{M_{i-1}}} & \mathbf{I_{m_i}} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{x}, & \mathbf{x_1^s}, & \ldots, & \mathbf{x_i^s}, & \ldots, & \mathbf{x_N^s}\end{bmatrix} = \mathbf{b_i}. \tag{4}$$

Next, each $\mathbb{M}_i$ generates a private random matrix $\mathbf{P_i} \in R^{k \times m_i}$, and left-multiplies both sides of Equation (4) to get:

$$\begin{bmatrix}\mathbf{P_i A_i} & \mathbf{0_{M_{i-1}}} & \mathbf{P_i} & \mathbf{0}\end{bmatrix}\begin{bmatrix}\mathbf{x}, & \mathbf{x_1^s}, & \ldots, & \mathbf{x_i^s}, & \ldots, & \mathbf{x_N^s}\end{bmatrix} = \mathbf{P_i b_i}. \tag{5}$$

Then each $\mathbb{M}_i$ adds

$$\begin{bmatrix}\mathbf{C_i^{|F|} - C_{i-1}^{|F|}} & \mathbf{0_{M_{i-2}}} & -\mathbf{C_{i-1}^{m_{i-1}}} & -\mathbf{C_{i-1}^{m_i} + C_i^{m_i}} & \mathbf{C_i^{m_{i+1}}} & \mathbf{0}\end{bmatrix},$$

to the coefficient matrix of the left-hand-side (LHS) of Equation (5), and adds $-\mathbf{D_{i-1}} + \mathbf{D_i}$ to its right-hand-side (RHS) to get Equation (6) where it can be observed that for each $\mathbb{M}_i$,

$$\left[\mathbf{P_i A_i + C_i^{|F|} - C_{i-1}^{|F|} \quad 0_{M_{i-2}} \quad -C_{i-1}^{m_{i-1}} \quad P_i - C_{i-1}^{m_i} + C_i^{m_i} \quad C_i^{m_{i+1}} \quad 0}\right] \cdot \left[\mathbf{x,\ x_1^s,\ \dots,\ x_i^s,\ \dots,\ x_N^s}\right] = \mathbf{P_i b_i - D_{i-1} + D_i}, \qquad (6)$$

the coefficient matrix of LHS of Equation (6) is of dimension k-by-$|F| + M_N$, and the RHS is of dimension k-by-1.

In the *transmission phase*, each $\mathbb{M}_i$ encrypts the set of linear equations in Equation (6) using a symmetric-key algorithm, *e.g.*, AES, and sends the cypher text to the aggregator. After collecting the linear equations from all member networks, the aggregator decrypts them and computes the sum of all LHS matrices and RHS matrices of all member networks, respectively. After simple elimination, the LHS sum is expressed as: $[\sum \mathbf{P_i A_i} \quad \mathbf{P_1} \quad \dots \quad \mathbf{P_N}]$. Similarly, the sum of all RHS matrices of all member networks can be expressed as $\sum \mathbf{P_i b_i}$. Denoting $[\mathbf{x_1^s}, \dots, \mathbf{x_N^s}]$ as $\mathbf{x^s}$, the aggregator can get the privacy-preserving abstraction $\Pi_p(F)$:

$$[\sum \mathbf{P_i A_i} \quad \mathbf{P_1} \quad \dots \quad \mathbf{P_N}.]\ [\mathbf{x},\ \mathbf{x^s}] = \sum \mathbf{P_i b_i}. \qquad (7)$$

Example: We use the example in Fig. 5 to illustrate the resource abstraction obfuscating protocol. For simplicity, we assume three member networks agree on $k = 4$. The private random matrices $\mathbf{P_1}$, $\mathbf{P_2}$ and $\mathbf{P_3}$ are generated as $\mathbf{P_1} = [11, 49, 95, 34]$, $\mathbf{P_2} = [58, 22, 75, 25]$, and $\mathbf{P_3} = [50, 69, 89, 95]$. The resource abstraction $\Pi_p(F)$ obtained by the aggregator is:

$$\begin{gathered}
69x_1 + 61x_2 + 11x_{11}^s + 58x_{21}^s + 50x_{31}^s = 4340, \\
71x_1 + 118x_2 + 49x_{11}^s + 22x_{21}^s + 69x_{31}^s = 7630, \\
170x_1 + 184x_2 + 95x_{11}^s + 75x_{21}^s + 89x_{31}^s = 14420, \\
59x_1 + 129x_2 + 34x_{11}^s + 25x_{21}^s + 95x_{31}^s = 7000,
\end{gathered}$$

where x_{11}^s, x_{21}^s and x_{31}^s are slack variables. Assume the user's objective is to maximize the throughput, *i.e.*, $x_1 + x_2$. Using this set of linear inequalities as the constraint, it can get the optimal solution where $x_1 = x_2 = 30$ Gbps, the same as when using Equation (2) as the constraint.

We conduct rigorous analysis on different properties (*e.g.*, correctness, security and efficiency) of the proposed obfuscating protocol, which can be found in Appendix B.

V. Super-Set Resource Abstraction Projection

One concern of the resource discovery is its scalability, as the number of resource discovery requests may be large in collaboration networks and each request would trigger a resource discovery procedure. This procedure requires the communication between the aggregator and the user, and between the aggregator and every Mercator domain server in member networks. Furthermore, the introduction of resource abstraction obfuscating further increases the communication and computation overhead of resource discovery. To address this issue, we develop a novel super-set projection technique. We describe its basic idea in this section, and leave the details of this technique in Appendix C.

Basic idea: The intuition of super-set projection is simple: to have the aggregator proactively discover the resource abstraction for a set of circuits between every pair of source and destination member networks, and use these pre-computed abstractions to quickly project to get the resource abstraction for user's requests.

In particular, in a collaboration network of N member networks, the super-set projection technique first simulates the need of $N(N-1)$ artificial circuits, where each circuit f_{ij} represents an artificial circuit from $\mathbb{M}_i$ to $\mathbb{M}_j$. With this artificial resource discovery request, the aggregator follows the normal resource discovery process to discover the shared bandwidth of all these $N(N-1)$ circuits across the whole collaboration network, represented by Π_{full}. When a user sends an actual resource discovery request for a set of F circuits, the aggregator checks the source and destination member networks of each circuit, and uses the stored Π_{full} to derive $\Pi(F)$ by removing unrelated inequalities and unrelated artificial circuits, instead of starting a new resource discovery procedure. In this way, the overhead of resource discovery is reduced to a single round of message exchange between the aggregator and the user.

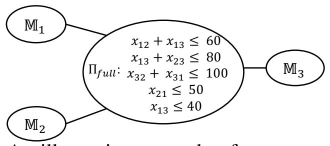


Fig. 7: An illustrating example of super-set projection.

Example: Consider an example of 3 member networks in Fig. 7. With the super-set projection, the aggregator discovers the bandwidth sharing of all $3 \times 2 = 6$ network-to-network artificial circuits as Π_{full} in the figure. When a user submits a resource discovery request for two circuits (S_1, D_1) and (S_2, D_2), where S_1 is in $\mathbb{M}_1$, S_2 and D_1 are in $\mathbb{M}_2$ and D_2 is in $\mathbb{M}_3$. The aggregator first maps the (S_1, D_1) to the artificial circuit from $\mathbb{M}_1$ to $\mathbb{M}_2$, and (S_2, D_2) to the artificial circuit from $\mathbb{M}_2$ to $\mathbb{M}_3$. Next, it projects Π_{full} to these two circuits to get the resource abstraction for these two circuits by (1) removing all linear inequalities that do not contain x_{12} or x_{23}, and (2) for every remaining linear inequality, remove all the items on the LHS that are not x_{12} or x_{23}. Finally, it returns the resource abstraction: $\{x_{12} \leq 60,\ x_{23} \leq 80\}$, to the user.

VI. Evaluation

We implement Mercator on commodity servers (*i.e.*, equipped with Intel(R) Xeon(R) E5-2609 2.50GHz 4-core CPU and 32 GB memory) and evaluate its performance based on a member-network-level topology from a large federation of networks supporting large-scale distributed science collaborations, and using real traffic traces from recent science experiments. After describing our experimental setup, we first demonstrate the benefits of resource abstraction through algebraic-expression enumeration. Second, we demonstrate the efficiency of the proposed resource abstraction obfuscation protocol. Finally, we demonstrate that the super-set projection technique substantially increases the scalability of Mercator.

A. Experimental Setup

We evaluate Mercator on the member-network-level topology from LHC Open Network Environment (LHCONE), a global science network consisting of 62 member networks, where scientists conduct large-scale distributed analytics. Because inter-member-network routing typically is not based on shortest path routing, but follows business relationships (*e.g.*, customer, peer, provider), we label the connections between every pair of connected member networks with their business relationship using the CAIDA network relationships dataset [40], and we compute the inter-member-network paths according to conventional policies for selecting and exporting routes. For member networks' intradomain topologies, we randomly select a topology for each network from the Topology Zoo [41], which provides a collection of real intradomain topologies. The topology of transit member networks varies from 31 switches/routers with 33 links to 49 switches/routers with 85 links. The topology of stub member networks (*e.g.*, campus science networks) ranges from 7 switches/routers with 6 links to 21 switches/routers with 44 links.

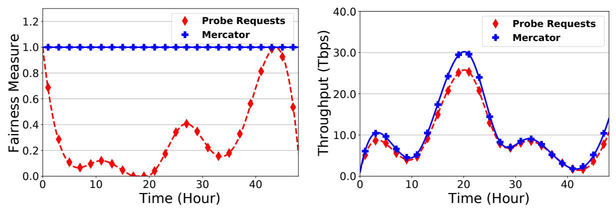
B. Benefits of Resource Abstraction Through Algebraic-Expression Enumeration

The first set of experiments demonstrate the benefits of the resource abstraction through algebraic-expression enumeration. We show that this abstraction reduces the time to discover network resources by up to 3 orders of magnitude, and allows fairer allocations of network resources.

1) Methodology: To evaluate the benefits of this resource abstraction, we replay the trace from a large-scale distributed experiment, and submit network resource reservations for the corresponding flows. More specifically, we use the actual trace from the CMS experiment [42], a major scientific experiment in LHC, and a main source of traffic in LHCONE. We extract the traffic flows, with their source member network, destination member network and the time. We focus on the 48-hour trace starting from December 14, 2017 and slice the data trace into 24 continuous 2-hour time windows. We apply the resources reservation once every time window. In other words, resources for traffic flows starting at the same time window are reserved in the same request, and we assume all resources will be released in the next time window.

We compare the performance of Mercator with that of existing reservation systems. In particular, for existing systems, we consider one that adopts a probe-requests based approach:

- **Mercator**: As described in Section II, for every resource discovery request, the aggregator queries the relevant member networks for their resource abstraction, and then derives the feasible bandwidth allocation region.
- **Probe requests**: As described in Section I, existing resource reservation systems such as OSCARS process each circuit in the request one at a time and in a sequential order. For each circuit, the resource reservation system initiates a depth-first search to probe if each member network can provide the requested bandwidth. We set the initial requested bandwidth for a circuit as C/N where C is the source host's capacity, and N is the number of flows from that host. In the event of a failure, the resource reservation system performs a binary search of the available bandwidth repeatedly halving the requested bandwidth until success. The process is repeated for each circuit in the request.



(a) Obj. = max-min fairness **(b)** Obj. = max total throughput

Fig. 8: Comparison of performance between the probe-requests approach and Mercator in different objectives.

2) Results: First, we consider that the goal of the resource allocation policy is to maximize the minimum throughput of all the requested flows (max-min fairness). Such a policy is commonly desired as it ensures high throughput and fairness across the circuits. We compare the fairness of the network resource allocations obtained with Mercator to that obtained with the probe-requests based solution. We adopt Jain's fairness index [43] to measure the fairness [31]:

$$J(x_1, x_2, \ldots, x_n) = \frac{(\sum_{i=1}^{n} x_i)^2}{n \cdot \sum_{i=1}^{n} {x_i}^2}$$

where x_i is the ratio of the actual allocation and the optimal fair allocation for a single flow. Fig. 8a shows that with resource abstraction, Mercator can always compute the optimal max-min fairness allocation. Hence its fairness index is always 1. In contrast, the fairness index of the probe-requests based solution has an average of 0.37, and even drops to 0 at times.

Second, we consider the case where the objective is to maximize the total throughput. Fig. 8b shows that the total throughput of Mercator is larger than that obtained by the probe-requests based solution, by 15% on average, and up to 20%. The results are noteworthy given that Mercator assumes the routes for each circuit to be completely determined by the underlying intradomain routing protocol. In contrast, the probe-requests approach sequentially explores every possible route for each circuit until it finds an available one. In other words, even with much less exploration, Mercator outperforms the probe requests. Allowing Mercator to consider not only the routes provided by the underlying routing protocols, but also all other available routes, could lead to significant additional improvements. We leave the extension of Mercator to consider all possible routes in the network as future work.

Fig. 9 presents the total resource discovery latency for completing all circuits resource reservations in a time window. We assume the aggregator to be in New York, and consider network latencies as measured in [44]. The figure shows the total resource discovery latency with Mercator can reduce the time to discover network resources by two orders of magnitude on average and up to three orders of magnitude at times. This is because resource abstraction allows users to query the information from different member networks in parallel. In contrast, existing probe-requests based solutions process requests sequentially, and continuously probe to discover the available network resources.

Finally, we highlight that the probe-requests based solution can suffer high request failure ratio, *i.e.*, a large number of requests cannot succeed: We define a failure of a request as the

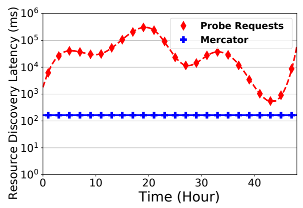


Fig. 9: Resource discovery latency of the probe-requests approach and Mercator.

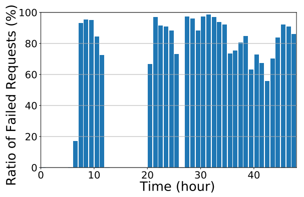
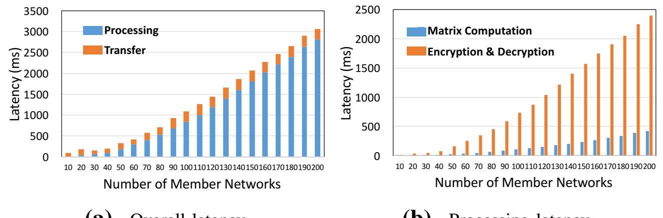


Fig. 10: Ratio of failed requests in the probe-requests approach.



(a) Overall latency. **(b)** Processing latency.

Fig. 11: The latency of the resource abstraction obfuscating protocol.

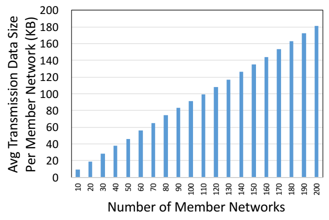


Fig. 12: The data transmission overhead of the resource abstraction obfuscating protocol.

inability to reserve resource for the circuit, due to the lack of remaining capacity despite the gradually decreasing requested bandwidth. Fig. 10 shows that during the 48-hour period Mercator is running, the probe-requests based solution has an average request failure ratio of 73%. In other words, more than 70% of the circuits cannot reserve network resources. This is because the probe-requests approach processes the request for each circuit sequentially. Therefore, the first few circuits may successfully reserve network resources and saturate the network. As such, despite achieving a total throughput similar to Mercator, the majority of the latter requests may fail as the links do not have any spare resources. In contrast, the request failure ratio of Mercator is null because Mercator returns a feasible region for the set of circuits so that the user can make optimal reservation decisions for all circuits.

C. Efficiency of Resource Abstraction Obfuscating Protocol

This second set of experiments evaluate the performance of the resource abstraction obfuscating protocol. We show that this protocol efficiently scales for collaboration networks of 200 member networks, with a maximal overall latency around 3 seconds and an average data transmission overhead between the aggregator and member networks of only around 180 KB.

1) Methodology: We conduct our experiment by using the member-network-level topology from the LHC Open Network Environment (LHCONE). In each round of the experiment, we randomly select a set of member networks from the topology. For each chosen member network, we randomly select a set of m linear inequalities, where m is randomly chosen between 5 and 15, to represent the bandwidth feasible reason for 10 circuits in this member network. For the encryption and decryption operations in the obfuscating protocol, we use the AES algorithm, provided by the Python Cryptography Toolkit (pycrypto) [45]. The parameters k, $\mathbf{C_i}$ and $\mathbf{D_i}$ are pre-configured as discussed in Appendix B.

We consider two metrics, *i.e.*, the latency and the data transmission overhead of the resource abstraction obfuscating protocol. First, the overall latency of the protocol is measured from the beginning of the obfuscation phase, when each member network independently starts to obfuscate its own set of linear inequalities, to the end of the transmission process, when the aggregator obtains $\sum \mathbf{P_i A_i x} = \mathbf{b}$. We use the field statistic results measured in [44] as the communication latencies between the aggregator and the Mercator domain servers at the different member networks. Second, the data transmission overhead is measured as the size of the set of encrypted, obfuscated linear equations transferred from each member network to the aggregator. We vary the number of member networks from 10 to 200, in a step size of 10. For each number of member networks, we repeat the experiment 10 times and measure the average values of these metrics.

2) Results: We present the results of our experiments in Fig. 11 and Fig. 12. In particular, Fig. 11a shows the overall latency of the obfuscating protocol under different numbers of member networks, together with a break down on processing delay and transmission latency. We observe that even for a large collaboration network with 200 member networks, which is larger than most existing operational collaboration networks, the overall latency of the resource abstraction obfuscating protocol is only slightly over 3 seconds, which demonstrates that the latency of this protocol is reasonably low. We also observe that the processing latency takes a much higher percentage than the transmission latency and that the processing latency has a linear growth as the number of member networks increases. We further plot the breakdown of the processing latency. Fig. 11b shows that both the cryptography operations of AES and the matrix operations in the resource abstraction obfuscating protocol increases linearly as the number of member networks increases, but the AES encryption and decryption operations are the most expensive operations in the protocol (*i.e.*, up to 2.4 seconds for federations of 200 member networks). More importantly, although the obfuscating protocol may take over 3 seconds for a federation of 200 member networks, we emphasize that with the super-set projection technique, the Mercator domain servers do not need to execute the obfuscating protocol for each individual request.

Next, we present the average data transmission overhead of the obfuscating protocol at each member network in Fig. 12. We see in this figure that even after the encryption, the size of data to be transmitted from member networks to the aggregator is still very small. For example, for a collaboration network with 200 member networks, the average size of data transmitted from a member network to the aggregator is only 180 KB. As discussed in Appendix B, this is because most of the columns of the LHS coefficient matrix are zero-columns and each member network only needs to send nonzero-columns to the aggregator. The linear scaling of the data transmission overhead (*i.e.*, the ciphertext) at each member network comes from the linear increase of the number of disguised linear equations (*i.e.*, the plaintext), which is caused by the linear increase of k due to the increased number of member networks. This is consistent with Proposition 3 in Appendix B.

D. Efficiency of Super-Set Projection

In this experiment, we evaluate the efficiency of the super-set projection technique in improving the scalability of Mercator. We show that this mechanism improves the resource discovery delay of Mercator by 2 times, and that its update latency is within seconds in a collaborative network with 200 member networks.

1) Methodology: We conduct our experiments by using the same settings as in Section VI-B1. We focus on two metrics. The first one is the resource discovery latency. When Mercator uses super-set projection, the resource discovery latency is reduced to only the round-trip time from the user to the Mercator aggregator because the aggregator can derive the resource abstraction for a request from the precomputed Π_{full}.

To have a comprehensive understanding on the scalability of super-set projection, we are also interested in a second metric, the update latency. This is measured as the resource discovery latency of from the time the aggregator starting the artificial resource abstraction discovery procedure to the time the aggregator receives the latest Π_{full}. In particular, we measure this latency under different collaboration scales by varying the number of member networks and the number of stub member networks in the collaborative network. For each setting, we repeat the experiment 10 times and compute the average update latency. In each repetition, we also randomly choose different sizes of intradomain topologies from the Topology Zoo dataset for each member network.

2) Results: Fig. 13 compares the resource discovery latency of Mercator with and without super-set projection. We observe that the super-set projection technique decreases the average resource discovery latency by around 2 times. Fig. 14 presents the update latency of this mechanism. It shows that even in a collaborative network with 200 member networks, the update latency of Π_{full} is still less than 10 seconds. Most importantly, although computing Π_{full} may take up to ten seconds for a federation of 200 member networks, we emphasize that resource discovery requests do not get blocked at the aggregator because servers from the aggregator pool can still process incoming requests using the previously computed resource abstraction, which is continuously locally updated (*e.g.*, available resources are continuously reduced as incoming requests reserve resources).

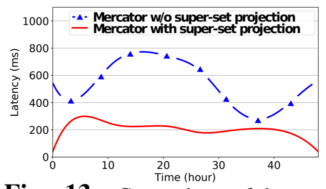


Fig. 13: Comparison of latency between Mercator with and without super-set projection.

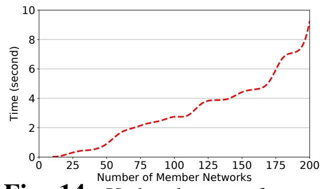


Fig. 14: Update latency of super-set projection.

VII. Related Work

Many multi-domain network resource information and reservation systems [4], [5], [8]–[10], [46] have been developed to support collaborative data sciences. Multiple multi-domain resource discovery systems (*e.g.*, [11]–[17]) are also designed to discover endpoint resources (*i.e.*, computation and storage resources) and their availability for different services across multiple domains. In contrast, there has been little progress on multi-domain network resource discovery systems that provide fine-grained, global network resource information, to support high-performance, collaborative data sciences.

Many cluster / grid resource management systems [15], [17], [27], [28], [47]–[53] adopt a graph-based abstraction to discover and manage network resources. However, in a multi-domain collaborative network, this abstraction would reveal the network topology and link availabilities of member networks, leading to security breaches. Some systems [29], [30] use a one-big-switch abstraction to provide a simplified view of network resources, which protects the privacy of member networks but cannot provide accurate information of shared network resource for concurrent traffic flows. Some recent studies [26], [33], [34], [54], [55] propose variations of the one-big-switch abstraction to represent the resource availability and sharing among different data traffic flows using operations defined on different algebra fields. However, this abstraction (1) cannot handle complex routing and traffic engineering policies, *e.g.*, WCMP, and (2) will raise security concern when applied to multi-domain science collaborations. In contrast, Mercator provides fine-grained, global network resource information, to support high-performance, collaborative data sciences, through a unifying representation and composition framework to reveal compact, complete multi-domain network resource information.

VIII. Conclusion

We present Mercator, a novel multi-domain network resource discovery system to provide fine-grained, global network resource information, to support high-performance, collaborative data sciences. The core of Mercator is a unifying representation resource abstraction using algebraic expressions to capture multi-domain network available bandwidth. We develop a resource abstraction obfuscating protocol and a super-set projection technique to ensure the privacy-preserving and the scalability of Mercator. Evaluation using real data demonstrates the efficiency and efficacy of Mercator.

Acknowledgment

The authors thank Kai Gao, Geng Li, Linghe Kong, Ennan Zhai, Alan Liu, Yeon-sup Lim and Haizhou Du for their help during the preparation of this paper. The authors also thank the anonymous reviewers for their valuable comments. This research is supported in part by NSFC grants #61702373, #61672385 and #61701347; China Postdoctoral Science Foundation #2017-M611618; NSF awards #1440745, #1246133, #1341024, #1120138, and #1659403; DOE award #DE-AC02-07CH11359; DOE/ASCR project #000219898; Google Research Award, and the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001.

References

[1] "The Large Hadron Collider (LHC) Experiment," https://home.cern/topics/large-hadron-collider.

[2] "The Square Kilometre Array," https://www.skatelescope.org/.

[3] "The Linac Coherent Light Source," https://lcls.slac.stanford.edu/.

[4] "Oscars: On-demand secure circuits and advance reservation system," https://www.es.net/engineering-services/oscars/.

[5] M. Campanella, R. Krzywania, V. Reijs, D. Wilson, A. Sevasti, K. Stamos, and C. Tziouvaras, "Bandwidth on demand services for european research and education networks," in Bandwidth on Demand, 2006 1st IEEE International Workshop on. IEEE, 2006, pp. 65–72.

[6] C. Guok, E. N. Engineer, and D. Robertson, "Esnet on-demand secure circuits and advance reservation system (oscars)," Internet2 Joint, vol. 92, 2006.

[7] W. Johnston, C. Guok, and E. Chaniotakis, "Motivation, design, deployment and evolution of a guaranteed bandwidth network service," in Proceedings of the TERENA Networking Conference, 2011.

[8] B. Riddle, "Bruw: A bandwidth reservation system to support end-user work," in TERENA Networking Conference, Poznan, Poland, 2005.

[9] J. Sobieski, T. Lehman, and B. Jabbari, "Dragon: Dynamic resource allocation via gmpls optical networks," in MCNC Optical Control Planes Workshop, Chicago, Illinois, 2004.

[10] X. Zheng, M. Veeraraghavan, N. S. Rao, Q. Wu, and M. Zhu, "Cheetah: Circuit-switched high-speed end-to-end transport architecture testbed," IEEE Communications Magazine, vol. 43, no. 8, pp. S11–S17, 2005.

[11] Y. Deng, F. Wang, and A. Ciura, "Ant colony optimization inspired resource discovery in p2p grid systems," The Journal of Supercomputing, vol. 49, no. 1, pp. 4–21, 2009.

[12] S. Fitzgerald, I. Foster, C. Kesselman, G. Von Laszewski, W. Smith, and S. Tuecke, "A directory service for configuring high-performance distributed computations," in IEEE HPDC 1997.

[13] A. Iamnitchi and I. Foster, "A peer-to-peer approach to resource location in grid environments," in Grid resource management. Springer, 2004, pp. 413–429.

[14] T. Kocak and D. Lacks, "Design and analysis of a distributed grid resource discovery protocol," Cluster Computing, vol. 15, no. 1, pp. 37–52, 2012.

[15] I. Sfiligoi, D. C. Bradley, B. Holzman, P. Mhashilkar, S. Padhi, and F. Wurthwein, "The pilot way to grid resources using glideinWMS," in CSIE. IEEE, 2009, pp. 428–432.

[16] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan, "Chord: a scalable peer-to-peer lookup protocol for internet applications," IEEE/ACM Transactions on Networking (TON), vol. 11, no. 1, pp. 17–32, 2003.

[17] D. Thain, T. Tannenbaum, and M. Livny, "Distributed computing in practice: the Condor experience," Concurrency and computation: practice and experience, vol. 17, no. 2-4, pp. 323–356, 2005.

[18] R. Ahmed, N. Limam, J. Xiao, Y. Iraqi, and R. Boutaba, "Resource and service discovery in large-scale multi-domain networks," IEEE Communications Surveys & Tutorials, vol. 9, no. 4, pp. 2–30, 2007.

[19] A. Hameurlain, D. Cokuslu, and K. Erciyes, "Resource discovery in grid systems: a survey," International Journal of Metadata, Semantics and Ontologies, vol. 5, no. 3, pp. 251–263, 2010.

[20] N. J. Navimipour, A. M. Rahmani, A. H. Navin, and M. Hosseinzadeh, "Resource discovery mechanisms in grid systems: A survey," Journal of Network and Computer Applications, vol. 41, pp. 389–410, 2014.

[21] S. Tuecke, V. Welch, D. Engert, L. Pearlman, and M. Thompson, "Internet x. 509 public key infrastructure (pki) proxy certificate profile," Tech. Rep., 2004.

[22] N. Sakimura, J. Bradley, M. Jones, and B. de Medeiros, "C. mortimore," openid connect core 1.0", november 2014."

[23] O. S. S. T. Committee et al., "Security assertion markup language (saml) 2.0," ht tp://www. oasis-open. org/committees/tc home. php, 2012.

[24] Y. Rekhter, S. Hares, and D. T. Li, "A Border Gateway Protocol 4 (BGP-4)," RFC 4271, Jan. 2006. [Online]. Available: https://rfc-editor.org/rfc/rfc4271.txt

[25] "Route views project," http://www.routeviews.org/routeviews/.

[26] V. Heorhiadi, M. K. Reiter, and V. Sekar, "Simplifying software-defined network optimization using sol." in NSDI, 2016, pp. 223–237.

[27] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, "Mesos: A platform for fine-grained resource sharing in the data center," in NSDI, 2011.

[28] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, "Large-scale cluster management at Google with Borg," in EuroSys. ACM, 2015, p. 18.

[29] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4p:provider portal for applications," Acm Sigcomm Aug, vol. 38, no. 4, pp. 351–362, 2008.

[30] R. Alimi, Y. Yang, and R. Penno, "Application-layer traffic optimization (ALTO) protocol."

[31] J. Y. Boudec, "Rate adaptation, congestion control and fairness: A tutorial," Web Page, no. Oct, 2000.

[32] F. P. Miller, A. F. Vandome, and J. McBrewster, "Advanced encryption standard," 2009.

[33] K. Gao, C. Gu, Q. Xiang, X. Wang, Y. R. Yang, and J. Bi, "ORSAP: abstracting routing state on demand," in IEEE ICNP 2016.

[34] K. Gao, Q. Xiang, X. Wang, Y. R. Yang, and J. Bi, "Nova: Towards on-demand equivalent network view abstraction for network optimization," in ACM/IEEE IWQoS 2017, 2017.

[35] J. Telgen, "Identifying redundant constraints and implicit equalities in systems of linear constraints," Management Science, vol. 29, no. 10, pp. 1209–1222, 1983.

[36] M. Raykova, Secure Computation in Heterogeneous Environments: How to Bring Multiparty Computation Closer to Practice? Columbia University, 2012.

[37] A. C.-C. Yao, "How to generate and exchange secrets," in IEEE FOCS 1986.

[38] X. Feng and Z. Zhang, "The rank of a random matrix," Applied mathematics and computation, vol. 185, no. 1, pp. 689–694, 2007.

[39] O. L. Mangasarian, "Privacy-preserving horizontally partitioned linear programs," Optimization Letters, vol. 6, no. 3, pp. 431–436, 2012.

[40] "The CAIDA AS Relationships Dataset, 2016," http://www.caida.org/data/as-relationships/.

[41] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The internet topology zoo," vol. 29, no. 9, pp. 1765–1775.

[42] "CMS Task Monitoring," http://dashb-cms-job.cern.ch/.

[43] R. Jain, D.-M. Chiu, and W. R. Hawe, A quantitative measure of fairness and discrimination for resource allocation in shared computer system. Eastern Research Laboratory, Digital Equipment Corporation Hudson, MA, 1984, vol. 38.

[44] "Global Ping Statistics - WonderNetwork, 2018," https://wondernetwork.com/pings/.

[45] "Python Cryptography Toolkit," https://pypi.python.org/pypi/pycrypto.

[46] "Network service interface," https://redmine.ogf.org/projects/nsi-wg.

[47] "Under the hood: Scheduling MapReduce jobs more efficiently with Corona," http://on.fb.me/TxUsYN, [Online; accessed: 09-May-2017].

[48] E. Boutin, J. Ekanayake, W. Lin, B. Shi, J. Zhou, Z. Qian, M. Wu, and L. Zhou, "Apollo: Scalable and coordinated scheduling for cloud-scale computing," in OSDI, 2014, pp. 285–300.

[49] C.-C. Hung, L. Golubchik, and M. Yu, "Scheduling jobs across geo-distributed datacenters," in SoCC. ACM, 2015, pp. 111–124.

[50] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg, "Quincy:fair scheduling for distributed computing clusters," in IEEE International Conference on Recent Trends in Information Systems, 2009, pp. 261–276.

[51] Q. Pu, G. Ananthanarayanan, P. Bodik, S. Kandula, A. Akella, P. Bahl, and I. Stoica, "Low Latency Geo-distributed Data Analytics," in SIGCOMM. ACM, 2015, pp. 421–434.

[52] R. Viswanathan, G. Ananthanarayanan, and A. Akella, "Clarinet: Wan-aware optimization for analytics queries," in Usenix Conference on Operating Systems Design and Implementation, 2016, pp. 435–450.

[53] A. Vulimiri, C. Curino, B. Godfrey, K. Karanasos, and G. Varghese, "WANalytics: Analytics for a geo-distributed data-intensive world," in CIDR, 2015.

[54] Q. Xiang, S. Chen, K. Gao, H. Newman, I. Taylor, J. Zhang, and Y. R. Yang, "Unicorn: Unified resource orchestration for multi-domain, geo-distributed data analytics," in 2017 IEEE SmartWorld, DAIS Workshop.

[55] Q. Xiang, X. Wang, J. Zhang, H. Newman, Y. R. Yang, and Y. J. Liu, "Unicorn: Unified resource orchestration for multi-domain, geo-distributed data analytics," in INDIS Workshop. IEEE, 2017.

[56] Q. Xiang, J. J. Zhang, X. T. Wang, Y. J. Liu, C. Guok, F. Le, J. MacAuley, H. Newman, and Y. R. Yang, "Fine-grained, multi-domain network resource abstraction as a fundamental primitive to enable high-performance, collaborative data sciences," in Technical Report.

Appendix A
Use Cases

In this appendix, we show how resource abstraction handles three important use cases in collaborative data sciences.

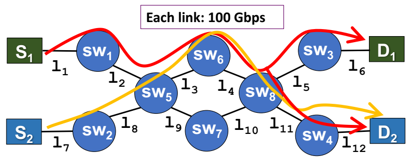


Fig. 15: A running example illustrating how the resource abstraction handles multicast through algebraic-expression enumeration, where two circuits $(S_1, \{D_1, D_2\})$ and (S_2, D_2) need to be reserved.

Use case 1 - multicast: Consider the example in Fig. 15, where the first circuit is a multicast circuit from S_1 to D_1 and D_2, and the second one is a unicast circuit from S_2 to D_2. The routes for these circuits, computed by the underlying routing protocol, are marked in red and yellow, respectively. The resource abstraction captures the bandwidth sharing between these two circuits by introducing auxiliary variables x_{11} and x_{12} for the multicast circuit. Because the traffic duplication for the first circuit happens at switch 8, we use x_{11} to represent the traffic from switch 8 to D_1, and x_{12} to represent the traffic from switch 8 to D_2. In this way, the Mercator domain server will generate the following set of linear inequalities:

$$\begin{aligned} & x_{11} = x_1,\ x_{12} = x_1, & \\ & x_1 \le 100 & \forall l_u \in \{l_1, l_2\}, \\ & x_{11} \le 100 & \forall l_u \in \{l_5, l_6\}, \\ & x_2 \le 100 & \forall l_u \in \{l_7, l_8\}, \\ & x_1 + x_2 \le 100 & \forall l_u \in \{l_3, l_4\}, \\ & x_{12} + x_2 \le 100 & \forall l_u \in \{l_{11}, l_{12}\}, \end{aligned} \tag{8}$$

Use case 2 - multi-path routing: Consider the example in Fig. 16, where the user wants to discover the bandwidth sharing for two circuits $f_1 : (S_1, D_1)$ and $f_2 : (S_2, D_2)$, and $\mathbb{M}_1$ uses multi-path routing for the circuit f_1, *i.e.*, routing to two egresses e_1, e_2.

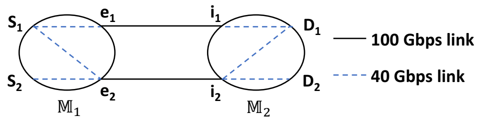


Fig. 16: A running example illustrating how resource abstraction handles complex routing and traffic engineering policies through algebraic-expression enumeration and how resource abstractions from different member networks are stitched, where two circuits (S_1, D_1) and (S_2, D_2) need to be reserved.

In particular, the Mercator domain server at $\mathbb{M}_1$ introduces variables x_{11} and x_{12} to represent the available bandwidth from S to egresses e_1 and e_2, respectively, and share the introduction of these variables to $\mathbb{M}_2$. Then $\mathbb{M}_1$ independently adds an equation $x_1 = x_{11} + x_{12}$ into its set of linear inequalities $\Pi_1(F)$. The resulting resource abstraction at both member networks are then expressed as

$$\begin{aligned} \Pi_1(F): \quad & x_1 = x_{11} + x_{12}, & \Pi_2(F): \quad & x_{11} \le 40, \\ & x_{11} \le 40, & & x_{12} \le 40, \\ & x_{12} \le 40, & & x_2 \le 40. \\ & x_2 \le 40, & & \\ & x_{11} \le 100, & & \\ & x_{12} + x_2 \le 100. & & \end{aligned} \tag{9}$$

Using $\Pi_1(F)$ and $\Pi_2(F)$ as the constraint, the user can then make reservation requests based on the optimization of her own objective function. For example, to achieve the max-min fairness between two circuits, the user will reserve $x_1 = 80$ Gbps for (S_1, D_1) and $x_2 = 40$ Gbps for (S_2, D_2), where internally $\mathbb{M}_1$ can allocate $x_{11} = x_{12} = 40$ Gbps.

Use case 3 - load-balancing: In the same example in Fig. 16, assume $\mathbb{M}_1$ uses weighted-cost-multi-path (WCMP) and has an internal policy to allocate bandwidth for the circuit (S_1, D_1) along two path $S_1 \to e_1$ and $S_1 \to e_2$ in a ratio of 1:2. With this policy, the previous reservation request with $x_1 = 80$ Gbps and $x_2 = 40$ Gbps is no longer valid as x_{11} and x_{12} cannot reach 40 Gbps simultaneously. To capture this policy so that the user does not make the invalid reservation request, the Mercator domain server at $\mathbb{M}_1$ introduces an additional equation $x_{12} = 2x_{11}$ into $\Pi_1(F)$ and sends to the user. And the user can compute the valid, optimal reservation decisions, *e.g.*, $x_1 = 60$ Gbps and $x_2 = 40$ Gbps, to achieve max-min fairness.

Appendix B
Analysis of the Resource Abstraction Obfuscating Protocol

In this appendix, we conduct rigorous analysis on different properties of the proposed obfuscating protocol.

Correctness: We first study the correctness of this protocol. In particular, we start by proposing and proving the following propositions.

***Proposition** 1 (Resource Abstraction Equivalence):* The bandwidth feasible region of the set of circuits F over N member networks represented by Equation (7) is the same as the bandwidth feasible region represented by $\mathbf{Ax} \le \mathbf{b}$ where $\mathbf{A} = [\mathbf{A_1}, \mathbf{A_2}, \dots, \mathbf{A_N}]$ and $\mathbf{b} = [\mathbf{b_1}, \mathbf{b_2}, \dots, \mathbf{b_N}]$.

Proof: To prove this proposition, we first observe that the bandwidth feasible region of $\mathbf{Ax} \le \mathbf{b}$ is the same as that of

$$\begin{bmatrix}\mathbf{A} & \mathbf{I_{M_N}}\end{bmatrix}\begin{bmatrix}\mathbf{x}, & \mathbf{x^s}\end{bmatrix} = \mathbf{b} \tag{10}$$

Representing $\mathbf{P} = [\mathbf{P_1}, \dots, \mathbf{P_N}] \in R^{k \times M_N}$, we first observe that $\begin{bmatrix}\sum \mathbf{P_i A_i} & \mathbf{P_1} & \dots & \mathbf{P_N}\end{bmatrix} = \mathbf{P}\begin{bmatrix}\mathbf{A} & \mathbf{I_{M_N}}\end{bmatrix}$, and that $\sum \mathbf{P_i b_i} = \mathbf{Pb}$ [39]. It is easy to see that when $\begin{bmatrix}\mathbf{x} & \mathbf{x^s}\end{bmatrix}$ satisfies Equation (10), it also satisfies Equation (7).

Next, from the results in [38] and that $\mathbf{P} \in R^{k \times M_N}$, we have $rank(\mathbf{P}) = M_N < k$. As a result, $\mathbf{P}$ has a left inverse matrix $\mathbf{P}_{left}^{-1}$ where $\mathbf{P}_{left}^{-1}\mathbf{P} = \mathbf{I}_{M_N}$. Hence when $\begin{bmatrix}\mathbf{x} & \mathbf{x^s}\end{bmatrix}$ satisfies Equation (7), *i.e.*, $\mathbf{P}\begin{bmatrix}\mathbf{A} & \mathbf{I_{M_N}}\end{bmatrix}\begin{bmatrix}\mathbf{x}, & \mathbf{x^s}\end{bmatrix} = \mathbf{Pb}$, we have

$$\mathbf{P}_{left}^{-1}\mathbf{P}\begin{bmatrix}\mathbf{A} & \mathbf{I_{M_N}}\end{bmatrix}\begin{bmatrix}\mathbf{x}, & \mathbf{x^s}\end{bmatrix} = \mathbf{P}_{left}^{-1}\mathbf{Pb},$$

which then transforms into Equation (10). Therefore, Equations (7) and (10) represent the same bandwidth feasible region, which completes the proof. ■

In addition, we also have an interesting lemma, which serves as an alternative correctness proof of the equivalence proposition.

***Lemma** 1:* Given the set of linear equations in Equation (7), the aggregator can use Gaussian Elimination to get $\mathbf{Ax} \le \mathbf{b}$. The complete proof can be found in our technical report [56].

Security: Next, we give the following proposition on the privacy-preserving property of the proposed protocol.

***Proposition** 2 (Resource Abstraction Privacy-Preserving):* In the semi-honest security model, the proposed resource abstraction obfuscating protocol ensures that (1) the aggregator cannot associate any linear equation it receives in $\Pi_p(F)$ with any particular member network, and (2) for any $\mathbb{M}_i$, it does not know any linear inequality from any other $\Pi_j(F_j)$ $(j \neq i)$.

The complete proof can be found in [56]. Even with Lemma 1 and the inter-member-network-path information of each circuit, the aggregator still cannot associate any linear inequality in $\mathbf{Ax} \leq \mathbf{b}$ with the corresponding member network or any networking device (*i.e.*, switch or link). This is because (1) the set of linear equations sent by each member network do not represent its original feasible region, and (2) the inter-member-network-path does not reveal any topology information inside member networks.

With both propositions, we can get the following theorem.

***Theorem** 1:* Given a set of circuits F that span over N member networks, the proposed resource abstraction obfuscating protocol ensures that the aggregator receives equivalent, privacy-preserving resource abstraction and each member network only knows its own bandwidth feasible region.

As stated in Section IV-A, the resource abstraction obfuscating protocol was designed for the semi-honest security model. In a malicious setting (*e.g.*, some member networks may collude or be breached by one same attacker), the colluded member networks or the attacker still cannot associate a linear inequality to any unbreached member network, as long as the aggregator is not breached.

Efficiency: We next analyze the efficiency of our protocol at different phases. During the initialization phase, the main overhead comes from the process each member network agreeing on k, and each $\mathbb{M}_i$ share $\mathbf{C_i}$ and $\mathbf{D_i}$ with $\mathbb{M}_{i+1}$. The first part can be efficiently realized using leader-election algorithms in ring topology or pre-configured. For the second part, it can be efficiently realized by sharing random seeds between $\mathbb{M}_i$ and $\mathbb{M}_{i+1}$. In the obfuscating phase, the computation overhead is also low because it only involves simple, cheap matrix operations, *e.g.*, addition and multiplication.

One may have concern on the transmission overhead of our protocol in the transmission phase because we disguise the set of linear inequalities of each member network into a larger set of linear equations. However, observing the set of equations sent by each $\mathbb{M}_i$ in Equation (6), we can see that most of the columns of the LHS coefficient matrix are zero-columns. Therefore, each $\mathbb{M}_i$ only needs to send nonzero-columns to the aggregator and specifies the indice of these columns, substantially reducing the amount of data needs to be transmitted from $\mathbb{M}_i$ to the aggregator. We quantify the transmission overhead of our obfuscating protocol as follows:

***Proposition** 3 (Transmission Overhead):* Given a resource discovery procedure for a set of circuits F spanning over N member networks, the transmission overhead of the resource abstraction obfuscating protocol at each member network is $O(k|F|)$, where $k > \sum m_i$.

Appendix C
Practical Issues of Super-Set Resource Abstraction Projection

In this appendix, we discuss practical issues of the super-set projection technique.

Update of Π_{full}: We ensure the freshness of Π_{full} via two mechanisms. First, the Mercator domain servers at member networks periodically send updated information to the aggregator. Second, when the reservation system receives and successfully executes a resource reservation request from the user, it sends a notification to the aggregator with the reservation details so that the aggregator can update Π_{full}. The aggregator will only query the Mercator domain servers to obtain an up-to-date abstraction for the user when the user fails to reserve the resource based on the projected abstraction.

Handling heterogeneous flows: One may notice that the super-set projection technique is designed based on the assumption that given a source-destination member network pair, all the traffic flows between these two member networks will be treated homogeneously by all other member networks. In practice, flows between the same source-destination member network pair may be handled differently by other member networks, *i.e.*, they are heterogeneous flows. To address this limitation, we use traffic classes to differentiate heterogeneous flows. In particular, for each source-destination member network pair with G different traffic classes, the super-set projection technique considers these classes as G separate artificial circuits and proactively discovers the bandwidth sharing among these G circuits and other artificial circuits.