NOVA: Towards On-Demand Equivalent Network View Abstraction for Network Optimization

Kai Gao[†‡], Qiao Xiang[§‡], Xin Wang[§‡], Yang Richard Yang[§‡] and Jun Bi[†*]
[§] Department of Computer Science, Tongji University
[†] Institute for Network Sciences and Cyberspace, Tsinghua University
[‡] Department of Computer Science, Yale University

***Abstract*—As many applications today migrate to distributed computing and cloud platforms, their user experience depends heavily on network performance. Software Defined Networking (SDN) makes it possible to obtain a global view of the network, introducing the new paradigm of developing adaptive applications with network views. A naive approach of realizing the paradigm, such as distributing the whole network view to applications, is not practical due to scalability and privacy concerns. Existing approaches providing network abstractions are limited to special cases, such as bottlenecks exist only at networks edges, resulting in potentially suboptimal or infeasible decisions. In this paper, we introduce a novel, on-demand network abstraction service that provides an abstract network view supporting not only accurate *end-to-end QoS metrics*, which satisfy the requirements of many peer-to-peer applications, but also *multi-flow correlation*, which is essential for bandwidth-sensitive applications containing many flows to conduct global network optimization. We prove that our abstract view is *equivalent* to the original network view, in the sense that applications can make the same optimal decision as with the complete information. Our evaluations demonstrate that the abstraction guarantees feasibility and optimality for network optimizations and protects the network service providers' privacy. Our evaluations also show that the service can be implemented efficiently; for example, for an extreme large network with 30,000 links and abstraction requests containing 3,000 flows, an abstract network view can be computed in less than one second.**

I. Introduction

Software Defined Networking (SDN) is a new emerging technique. It enables a network to collect information from all the devices and construct a global view. This global view makes it possible to share with applications essential quality of service (QoS) metrics such as available bandwidth, loss rate and routing cost values, which are critical to performance of network-based services.

While northbound APIs for "apps" (management programs) to access the global view have been provided by many SDN controllers (*e.g.*, [1]–[4]), they are not open to non-administrative parties such as content providers, neighbor domains and VPN tenants, because of privacy, security and consistency concerns. Therefore, a new network abstraction providing global network view for non-administrative parties is needed.

A naive design is to only return a *slice* of the network to these non-administrative parties, or as we refer to as network *consumers*. Specifically, the *slice* can be calculated by an SDN controller which receives a query containing the flows of interest from a consumer, calculates the forwarding paths for the flows, and returns only links on the paths with associated attributes. However, this simple approach can have drawbacks. First, a slice can reveal sensitive information like network topology, leading to privacy leaks. Second, a slice may contain redundant information and introduce unnecessary communication overhead.

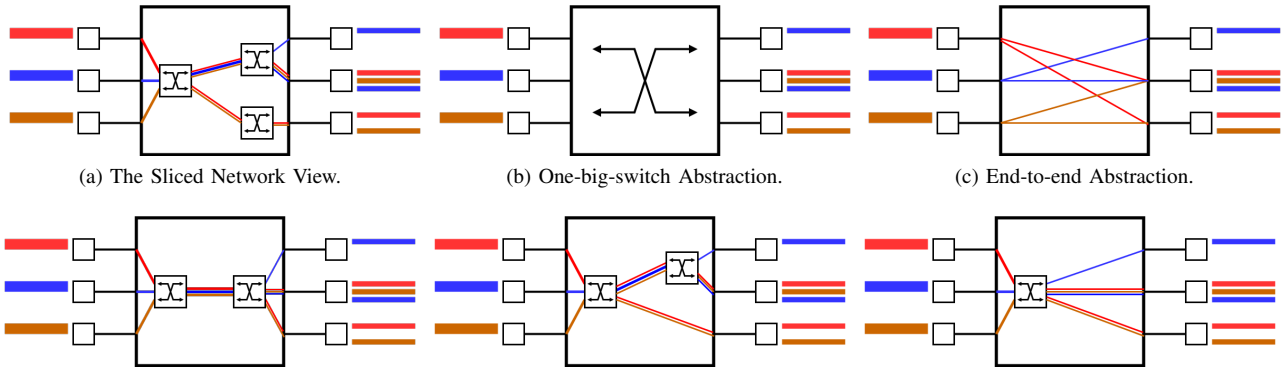
Thus, a problem arises on how to provide an abstract network view which can both eliminate these drawbacks and still provide high-quality information. It is non-trivial because of the following challenges:

- *Feasibility*: A decision made with the abstract view should also be feasible in the original network. Infeasible solutions such as flow rate scheduling can lead to congestion and significantly affect the total throughput and also the user experience.
- *Generality*: The abstraction should be general enough to provide fine-grained information and suffice the demands of applications with heterogeneous objectives.
- *Optimality*: A decision made with the abstract view should be as optimal as with the original network information. A suboptimal solution will affect the quality of service and cannot fully utilize the network resources.
- *Privacy*: The abstraction must be able to protect the privacy of the network service provider, making it difficult for malicious consumers to infer the original information.
- *Efficiency*: The abstraction should not introduce too much computation/communication overhead, even with moderately large networks and workloads.

Existing abstractions [5]–[14] usually target at a certain type of scenario and cannot support applications which require fine-grained QoS metrics. For example, many of these abstractions do not have the ability to accurately represent bottlenecks shared by multiple correlated flows in an arbitrary network, which is critical in emerging use cases such as geo-distributed data centers [15]–[17] and scientific computing platforms [18].

In this paper, we formally define the problem of providing high-quality on-demand abstract network views and make the first step by introducing the *equivalent network view* which guarantees *feasibility*, *generality* and *optimality*. The *equivalent network view* is based on the observation that these network information are eventually used by network

* The corresponding author is Prof. Jun Bi. junbi@tsinghua.edu.cn



(a) The Sliced Network View. (b) One-big-switch Abstraction. (c) End-to-end Abstraction.

(d) Incorrect Topology Aggregation. (e) Simple Equivalent Aggregation. (f) Advanced Equivalent Transformation.

There are 6 flows: 2 red, 2 blue and 2 brown. The flows are labeled as b_1, r_1, y_1, b_2, r_2 and y_2 from top to bottom.

Fig. 1: Comparison between Different Network View Abstractions.

consumers to conduct network optimizations.

Based on the concept of equivalent network view, we propose NOVA, the *Network Optimization View Abstraction*, which can effectively find such *equivalent network views*. Demonstrated by analysis and evaluations, NOVA can also achieve good *privacy preservation* and *efficiency*.

Our main contributions in this paper can be summarized as follows:

- We systematically investigate the problem of providing on-demand view abstraction for arbitrary application-layer network optimization, which is a missing functionality from current SDN northbound API design. We discuss the potential benefits for such a new functionality and identify its challenges.
- We propose NOVA to address the challenges, the *Network Optimization View Abstraction*, which is based on the concept of *equivalent network view*. We prove our equivalent transformation algorithms can effectively generate *equivalent network views* which can achieve feasibility, generality and optimality, while protecting the network privacy.
- We implement a prototype of NOVA and evaluate its performance using real topologies. Evaluations show that NOVA guarantees both feasibility and optimality, improves privacy by reducing information leak, and reduces the communication overhead by a factor of 1.25 to 5 even for large networks, for example, real ISP network topologies with up 10000 nodes and 30000 links, and large workloads (more than 3000 flow requests).

The rest of the paper is organized as follows. We summarize the demands for fine-grained network views, existing abstractions and their limitations in Section II. A formal description of the abstraction problem and the *equivalent network view* are then given in Section III, followed up by the algorithms in Section IV. We evaluate the prototype and analyze the results in Section V. Finally, we discuss the related work and give conclusions in Section VI and Section VII respectively.

II. Motivation

In this section we discuss the motivations that drive our research. See the network in Figure 1a, assume on the left-hand side, a web service provider has three services colored in red, blue and brown respectively while on the right-hand side there are six clients using different services. Assume the red service is live streaming, blue is video subscribing and brown is large file downloading. All three services require bandwidth so it's important to know about the bottlenecks in the network and the content provider sends a request on the bandwidth correlation of the six flows to the network. Meanwhile, the content provider does not want the network to know about how it would manage the services so it does not provide any further information.

The naive approach returns the slice containing all the links on the flow paths with the associated bandwidth information and how the links are shared by the flows, denoted by Figure 1a in this case. However, this can lead to information leaks so that malicious network consumers may leverage this service to infer the network information, which jeopardizes the privacy of the network provider. Also as the network size increases, the topology cannot be effectively represented.

The hose model, also known as the *one-big-switch* abstraction, returns the network as a single big switch as demonstrated in Figure 1b. However, the content provider would only know about the available bandwidth on the ingress/egress "port". If the bottleneck is the upper middle link, it is not propagated to the content provider. Thus, the content provider may incorrectly increase the traffic for the blue flows without knowing that they are correlated with r_1, which leads to congestion. Because of the TCP congestion control mechanism, congestion would not only affect the r_1, jeopardizing the overall objective to maximize the weighted quality of service, but also the throughput of the other flows sharing the same link, leading to an overall performance degradation.

The end-to-end abstraction as demonstrated in Figure 1c is barely useless in this case. It has the same problem as the *one-*

big-switch abstraction that information about the bottleneck within the network cannot be accurately provided to the consumer.

Topology aggregation [13] is a common technique in reducing the topology size. However, it also suffers from the incapability of providing accurate information about the flow correlations. What is worse, if not aggregated correctly as in Figure 1d, it may introduce unnecessary bottlenecks that lead to suboptimal decisions right in the beginning.

A simple observation is that the lower middle link and the lower right link are both shared by r_2 and y_2 only. Thus, we can aggregate them together as a new virtual link, as demonstrated in Figure 1e. One may think we can just delete one of them. However, if the content provider asks about the end-to-end routing metrics such as hop count at the same time, deletion would return incorrect values for the two flows.

At the same time, if we already know that the upper middle link would not be the bottleneck, the network view can be further reduced as demonstrated in Figure 1f. It is worth noticing that just like the case with the simple equivalent aggregation, we cannot just delete it if end-to-end metrics are also requested.

Thus, the question arises on how we can determine what kind of links can be reduced and how to reduce them correctly. In order to answer this question, we introduce the concept of *equivalent network view* and propose NOVA, the *Network Optimization View Abstraction*, with efficient algorithms to compute equivalent network view.

III. Equivalent Network View

In this section, we formally define *equivalent network view* after introducing its theoretical foundations – the variant routing metric algebra, and the unified network element. We prove its properties and demonstrate that it guarantees *generality*, *feasibility* and *optimality*.

A. The variant routing metric algebra

The routing metric algebra is introduced by Sobrinho to compute QoS-based routing [19]. The routing metric algebra is based on *path concatenation*. The routing metric algebra consists of the following operations: the *weight* function w, the concatenation operator $\circ$, the "plus" operator $\oplus$ and a binary relation $\preceq$.

The original routing metric algebra system can be represented as $(P, S, w, \circ, \oplus, \preceq)$. P is the set of paths and S is a closed set of metrics. The weight function w maps a path from P to a given metric in S. The concatenation operator $\circ$ is a binary operator on P, which takes two paths and returns a new one. Operator $\oplus$ is a binary operator on S and $\preceq$ is a binary relation on S.

In this paper, we introduce a variant of the routing metric algebra. First, to better formulate the constraints on *flow-independent* metrics, we introduce a new $\otimes : \mathbb{R} \times S \mapsto S$ operator to linearize the metric calculation. Second, we relax the constraint on path concatenation in [19], by extending the meaning of P from the set of paths to the set of *unified network elements* as defined in Section III-B. The new metric algebra can be described as $(P, S, w, \circ, \oplus, \preceq, \otimes)$. Concrete examples of some common routing metrics are demonstrated in Table I.

$(S, \oplus)$ is a semigroup so that $\oplus$ is *commutative* and *associative*. We also require that the $\otimes$ operator is *distributive* over $\oplus$. It can be easily proved that all the examples listed in Table I satisfy these requirements.

B. Unified network elements

Traditional graph representations of a network would treat links and nodes differently because the routing capability is only provided on nodes (switches/routers/middleboxes). However, since routing is not a mandatory functionality in our network view definition, we can generalize the nodes and links as *unified network elements* to simplify the representation.

For example, a deep packet inspection (DPI) middlebox may have a maximum processing speed, which yields the constraint on the total throughput passing through this DPI node. From the consumers' perspective, it has no difference as a shared bottleneck link. To guarantee that these unified network elements would not affect the results of routing metric algebra, we must alter the weight function w as:

$$w^*(p) = \begin{cases} w(p) & \text{if } w(p) \text{ exists} \\ e & \text{otherwise} \end{cases}$$

where e is the *identity* of w and some concrete examples are given in Table I.

Unified network elements have several benefits. First, this unified representation of links/nodes greatly simplifies our analysis. Second, it provides a high-level abstraction which focuses on the routing metric semantics and allows consumers to ignore how the metrics are computed/constrained.

C. Network view abstraction

We identify two kinds of routing metrics that are essential for network optimization:

- *Flow-independent* metrics represent those metrics whose value is *independent* of the flow correlations, in the sense that the existence of a flow would not affect the value of another flow's flow-independent metric sharing the same network element. Common flow-independent metrics used in QoS routing [20] include hop count, local link preference, delay, jitters and loss rate[1].
- The *flow-correlated* metrics represents the network resources that are shared among flows. Bandwidth is the most common and also the most important *flow-correlated* metric. Other shared resources such as flow entries or middlebox-related metrics may also exist in certain scenarios.

Flow-independent metrics are formulated as *equations* according to the routing metric algebra introduced in Section III-A. For example, the hop count between two end hosts is equal to the sum of the all hop count on each link in the

[1]Delay, jitters and loss rate are sensitive to traffic volumes so their real time values are not flow independent. However, they are considered flow independent when measured *statistically*.

TABLE I: The Variant Routing Metric Algebra.

S	Weight function (w)	$w(p_1)$	$w(p_2)$	$w(p_1 \circ p_2) = w(p1) \oplus w(p_2)$	$N \otimes w(p_1)$	Identity (e)	Zero ($\mathbf{0}$)
$\mathbb{N}^+$	Hopcount	h_1	h_2	$h_1 + h_2$	$N \cdot h_1$	0	$+\infty$
$\mathbb{R}^+$	Bandwidth	b_1	b_2	$\min(b_1, b_2)$	b_1	$+\infty$	0
$\mathbb{R}^+$	Delay	d_1	d_2	$d_1 + d_2$	$N \cdot d_1$	0	$+\infty$
$[0, 1]$	Loss rate	r_1	r_2	$1 - (1 - r_1)(1 - r_2)$	$1 - (1 - r_1)^N$	0	1

TABLE II: Symbols for Network View Abstraction.

Symbol	Meaning
V	Network view represented by 4-tuple
$V_i(\boldsymbol{u}_i)$	The i-the component (and its key matrix) of V
r_{ij}	Indicator of whether element j appears in flow i's path
$p_j^k(\hat{p}_i^k)$	k-th flow-independent metric of element j (flow i)
a_{ji}^k	Proportion of f_i's correlated metric k on element j
$q_j^k(\hat{q}_i^k)$	k-th flow-correlated metric of element j (flow i)
$\mathcal{E}$	Objective function of the consumer
$\mathcal{A}$	Abstraction function
V'	Abstract view computed by $\mathcal{A}(V)$

path. Let $\boldsymbol{p}_j$ represent the metrics on element j, $\hat{\boldsymbol{p}}_i$ represent the metrics for flow i, and $r_{ij}(r_{ij} \in \{0, 1\})$ represent whether element j appears in the path of flow i, we have:

$$\hat{\boldsymbol{p}}_i = \left(\bigoplus_j r_{ij} \otimes p_j^1, \ \ldots \ , \bigoplus_j r_{ij} \otimes p_j^s \right) = \boldsymbol{R}_i \times \boldsymbol{P}$$

If the k-th *flow-correlated* metric (resource) flow i consumes on element j is proportional to the total amount of resources consumed by this flow and the proportion is independent of flow correlations, the *flow-correlated* metrics are formulated as *linear constraints*. The sum of the resource consumption of all the flows on a single element must not exceed the available amount. Let $\boldsymbol{q}_j$ represent the available resources on element j, $\hat{\boldsymbol{q}}_i$ represent the resource consumed by flow i, and $a_{ji}^k (\geq 0)$ represent the proportion of the k-the kind of resource consumed by flow i on element j, we have:

$$\sum_i a_{ji}^k \hat{q}_i^k \leq q_j^k \Leftrightarrow \boldsymbol{A}^k \hat{\boldsymbol{q}}^k \leq \boldsymbol{q}^k$$

Thus, the network view can be formulated as a tuple with four elements: $V = (\boldsymbol{R}, \boldsymbol{P}, \boldsymbol{A}, \boldsymbol{Q})$. Based on this network view model, an *abstraction* can be defined as below:

Definition 1 (Network View Abstraction)**.** The network view abstraction is a transform function $\mathcal{A}$ which takes the original network view V and returns the *abstract* view V', i.e.,

$$V'(\boldsymbol{R'}, \boldsymbol{P'}, \boldsymbol{A'}, \boldsymbol{Q'}) = \mathcal{A}\left(V(\boldsymbol{R}, \boldsymbol{P}, \boldsymbol{A}, \boldsymbol{Q})\right)$$

D. Equivalent network view

A key insight of the network view abstraction is that the returned information is usually the input parameters of a specific algorithm, whose result can help applications make decisions. If the applications can make the same optimized decision with the abstract network view, we can say the abstract network view is *equivalent*.

Consider a consumer of the network view whose optimization problem consists of a set of flows $\{f_1, \ldots, f_n\}$. For flow i, the consumer queries its *flow-independent* metrics $\hat{\boldsymbol{p}}_i = (\hat{p}_i^1, \ldots, \hat{p}_i^s)$, and its *flow-correlated* metrics $\hat{\boldsymbol{q}}_i = (q_i^1, \ldots, q_i^t)$. The optimization problem can be formulated as

$$\begin{aligned} & \min \mathcal{E}\left(\{\hat{\boldsymbol{p}}_i\}, \{\hat{\boldsymbol{q}}_i\}\right) \\ & \hat{\boldsymbol{P}} = \boldsymbol{R} \times \boldsymbol{P} \\ \text{s.t.} \quad & \boldsymbol{A}^k \hat{\boldsymbol{q}}^k \leq \boldsymbol{q}^k \\ & \boldsymbol{R} \geq 0, \boldsymbol{A}^k \geq 0, \boldsymbol{q}^k \geq 0, \hat{\boldsymbol{q}}^k \geq 0 \end{aligned}$$

We use the symbols in Table II and define the equivalent network view as the following:

Definition 2 (Network View Equivalence)**.** Two network views V_1 and V_2, V_1 is *equivalent* to V_2 if and only if for any consumer with flows $\{f_i\}$, its objective function $\min \mathcal{E}\left(\{\hat{\boldsymbol{p}}_i\}, \{\hat{\boldsymbol{q}}_i\}\right)$ achieves the same optimal objective.

We use the symbol "$\sim$" to represent the network view equivalence. It can be easily proved that the network view equivalence is an *equivalence relation*. We also propose the criterion in Theorem 1 to simplify the verification.

Theorem 1 (Network View Equivalence Criterion)**.** Two network views V_1 and V_2, V_1 is *equivalent* to V_2 if and only if for any consumer with flows $\{f_i\}$ and metrics $\{\hat{\boldsymbol{p}}_i\}, \{\hat{\boldsymbol{q}}_i\}$:

$$\boldsymbol{R_1} \times \boldsymbol{P_1} = \boldsymbol{R_2} \times \boldsymbol{P_2} \tag{1a}$$

$$F_1^k = \left\{\boldsymbol{x} \mid \boldsymbol{A_1^k x} \leq \boldsymbol{q_1^k}\right\} = \left\{\boldsymbol{x} \mid \boldsymbol{A_2^k x} \leq \boldsymbol{q_2^k}\right\} = F_2^k \tag{1b}$$

Proof. **Sketch**: For two network views V_1 and V_2, we use $V_1 \sim^* V_2$ and $V_1 \sim V_2$ to represent that V_1 and V_2 are equivalent by the criterion and are equivalent by definition respectively. It can be easily proved that if $V_1 \sim^* V_2$, $V_1 \sim V_2$. For the other direction, we prove it by contradiction.

We first assume that there exists $V_1 \sim V_2$ but $V_1 \nsim^* V_2$, i.e., either Equation 1a or Equation 1b does not hold.

Assume Equation 1a does not hold. For any $\hat{\boldsymbol{P}}_\mathbf{1} = \boldsymbol{R_1} \times \boldsymbol{P_1} \neq \boldsymbol{R_2} \times \boldsymbol{P_2} = \hat{\boldsymbol{P}}_\mathbf{2}$, we find one entry that is not equal in $\hat{\boldsymbol{P}}_\mathbf{1}$ and $\hat{\boldsymbol{P}}_\mathbf{2}$, say $\hat{p_1}_i^k \neq \hat{p_2}_i^k$ and construct an objective function which use the $\hat{p}_i^k$ as the objective value. Thus, for the objective function the two network views yield different optimal objectives and they are not equivalent by definition, which contradicts with our assumption.

Assume Equation 1b does not hold, $\exists \boldsymbol{x_0} \in (F_1 \setminus F_2) \cup (F_2 \setminus F_1)$. Without loss of generality, let $\boldsymbol{x_0} \in F_1 \setminus F_2$. Since $\boldsymbol{x_0} \notin F_2$, there exist j, k such that $\boldsymbol{A_2}_j^k \boldsymbol{x_0} = y_0 > q_j^k$. Now

we construct a linear objective function in the standard form $\min -\boldsymbol{A_2}_j^k \boldsymbol{x}$, and assume the optimal objective is y_1 and y_2 respectively. We have $y_1 \geq y_0 > q_j^k = y_2$ which means the objective function yields different optimal objective values for the two network views. Again, we get a contradiction.

Thus, we can conclude that if $V_1 \sim V_2$, $V_1 \sim^* V_2$ and the criterion is both sufficient and necessary. □

We have proved that the network views satisfying this criterion also satisfy Definition 2 and can provide the *equivalent network view* for consumers with arbitrary objective functions and arbitrary fine-grained routing metrics as long as they fit in the *variant routing metric algebra*.

IV. Network Optimization View Abstraction

In this section, we introduce NOVA, the *Network Optimization View Abstraction* which conducts equivalent transformations to obtain the equivalent network view. NOVA consists of two algorithms, namely the *equivalent element aggregation* and the *equivalent element decomposition*. We prove both algorithms guarantee the *equivalence* condition, and analyze how they can improve *efficiency* and *privacy* as well.

To help simplify the analysis, we assume the request contains n flows with s flow-independent metrics and t flow-correlated metrics, while the corresponding original network view contains m elements.

A. Equivalent element aggregation

In this section, we introduce the *equivalence aggregation*. The intuition is that, as demonstrated with Figure 1e, the elements shared by the same set of flows and with the same coefficients in the network constraints can be combined into a single element. The algorithm is given in Algorithm 1 and we analyze its efficiency and prove its correctness.

Algorithm 1: NOVA Equivalent Element Aggregation

Input: $V(\boldsymbol{R}, \boldsymbol{P}, \boldsymbol{A}, \boldsymbol{S})$
Output: $V'(\boldsymbol{R'}, \boldsymbol{P'}, \boldsymbol{A'}, \boldsymbol{S'})$
1 **Function** EquivAggregation*(V)*
2 $\quad \mathcal{V} \leftarrow \left\{V_i \mid V_i \leftarrow \left(\boldsymbol{R^T}_i, \boldsymbol{P}_i, \{\boldsymbol{A}_i^k\}, \{\boldsymbol{Q}_i^k\}\right), 1 \leq i \leq m\right\}$
3 $\quad \mathcal{G} \leftarrow$ GroupBy$(\mathcal{V}, V_i \Rightarrow \left(k_i \leftarrow (\boldsymbol{R^T}_i, \{\boldsymbol{A}_i^k\}), V_i\right))$
4 $\quad$ **for** $G_i \in \mathcal{G}$ **do**
5 $\quad\quad V_i' \leftarrow$ Aggregate$(k_i, \{V_{a_j^i} \in G_i\})$
6 $\quad V' \leftarrow$
$$\left(\left[\boldsymbol{R'^T}_1 \cdots \boldsymbol{R'^T}_{m'}\right], \begin{bmatrix} \boldsymbol{p'}_1 \\ \vdots \\ \boldsymbol{p'}_{m'} \end{bmatrix}, \begin{Bmatrix} \boldsymbol{A'}_1^k \\ \vdots \\ \boldsymbol{A}_{m'}^k \end{Bmatrix}, \begin{Bmatrix} \boldsymbol{q'}_1^k \\ \vdots \\ \boldsymbol{q'}_{m'}^k \end{Bmatrix}\right)$$
$\quad$ **return** V'
7 **Function** Aggregate*(*$k_i, \{V_{a_j^i}\}$*)*
8 $\quad (\boldsymbol{R'^T}_i, \{\boldsymbol{A'}_i^k\}) \leftarrow k_i$
9 $\quad \boldsymbol{p'}_i \leftarrow [\bigoplus p_{a_j^i}^1, \ldots, \bigoplus p_{a_j^i}^s]$
10 $\quad \boldsymbol{q'}_i \leftarrow \left\{q_i'^k \mid q_i'^k \leftarrow \min\{q_{a_1^i}^k, \ldots, q_{a_t^i}^k\}, \forall k\right\}$
11 $\quad$ **return** $\left(\boldsymbol{R'^T}_i, \boldsymbol{p'}_i, \{\boldsymbol{A'}_i^k\}, \{\boldsymbol{q'}_i^k\}\right)$

The network view is represented as row vectors (components), as demonstrated in Line 2. Line 3 groups the i-th *component* $V_i(\boldsymbol{R}_i^T, \boldsymbol{P}_i, \{\boldsymbol{A}_i^k\}, \{\boldsymbol{Q}_i^k\})$ using a unified row vector $\boldsymbol{u}_i = \left[\boldsymbol{R^T}_i^T, \boldsymbol{A}_i^1, \ldots, \boldsymbol{A}_i^t\right]_{1\times(n+nt)}$ as the key. Line 5 computes the aggregation of the components in each group. Finally Line 6 constructs the new network view by merging all the aggregated components. For each component V_i, the time complexity for the grouping and the aggregation is $O(n(s+t))$ and $O(n(s+t))$ respectively while the Merge process is totally logical, which yields a total time of $O(mn(s+t))$.

Now we prove the element aggregation algorithm is correct, in the sense that it maintains the equivalence condition.

Theorem 2. $V' \leftarrow$ EquivAggregation(V), $V' \sim V$.

Proof. **Sketch**: Assume $\boldsymbol{a_i}$ represents the index of the components in G_i, and let $b_i \leftarrow \min a_{ij}$ and $c_i^k \leftarrow \arg\min_{j\in\boldsymbol{a_i}} q_j^k$.

First we check Equation 1a is met. Let $\boldsymbol{M} = \boldsymbol{R} \times \boldsymbol{P}$, we have

$$m_{ij} = \bigoplus_k r_{ik} \otimes p_k^j = \bigoplus_{1\leq k\leq m'} \left\{\bigoplus_{u\in\boldsymbol{a_k}} r_{iu} \otimes p_u^j\right\}$$
$$= \bigoplus_{1\leq k\leq m'} r_{ib_k} \otimes \left\{\bigoplus_{u\in\boldsymbol{a_k}} p_u^j\right\} = \bigoplus_{1\leq k\leq m} r_{ib_k} \otimes p'^j_u = m'_{ij}$$

The key steps are based on that $\oplus$ is transitive and commutative, $\otimes$ is distributive over $\oplus$, and $\forall u \in \boldsymbol{b}_k, r_{iu} = r_{ia_k}$.

Now we check Equation 1b. For any k, we have

$$F^k = \left\{\boldsymbol{x} \mid \boldsymbol{A^k}_i \boldsymbol{x} \leq \boldsymbol{q^k}_i, \forall i\right\}$$
$$F'^k = \left\{\boldsymbol{x} \mid \boldsymbol{A'^k}_i \boldsymbol{x} \leq \boldsymbol{q'^k}_i, \forall i\right\} = \left\{\boldsymbol{x} \mid \boldsymbol{A^k}_{c_i} \boldsymbol{x} \leq \boldsymbol{q}_{c_i}^k, \forall i\right\}$$

Since the constraints of F'^k is a subset of F^k, $F^k \subseteq F'^k$. If $F'^k \neq F^k$, $\exists \boldsymbol{x_0} \in F'^k \setminus F^k$, meaning $\boldsymbol{x_0}$ must at least violates one constraint in F^k, say $\boldsymbol{A^k}_{d_i}$ where $d_i \in \boldsymbol{a_i}$. Thus, we have

$$\boldsymbol{A^k}_{d_i} \boldsymbol{x_0} > q_{d_i}^k \geq \min_{j\in\boldsymbol{a_i}} q_j^k = q_{c_i}^k$$

which means $\boldsymbol{x_0}$ also violates one constraint in F'^k and leads to contradiction with our assumption. So we have $F^k = F'^k$.

By Theorem 1, $V' \sim V$. □

B. Equivalent element decomposition

In this section, we introduce the motivations for *equivalent element decomposition* which can substantially improve the performance of *equivalent element aggregation*.

Algorithm 1 guarantees the equivalence condition which is important to prove the correctness of NOVA, however, the condition to aggregate components is not easy to be satisfied without further processing. Thus, in practice we need to conduct another equivalent transformation, namely *equivalent element decomposition*. The intuition of this algorithm can be demonstrated using the simple example below:

$$a : \textit{routingcost} = 1, \textit{bandwidth} = 100\text{Mbps}$$
$$b : \textit{routingcost} = 2, \textit{bandwidth} = 100\text{Mbps}$$
$$c : \textit{routingcost} = 3, \textit{bandwidth} = 200\text{Mbps}$$
$$\boldsymbol{R}_a^T = \boldsymbol{A}_a = \begin{bmatrix}1 & 0\end{bmatrix}^T, \boldsymbol{R}_b^T = \boldsymbol{A}_b = \begin{bmatrix}0 & 1\end{bmatrix}^T, \boldsymbol{R}_c^T = \boldsymbol{A}_c = \begin{bmatrix}1 & 1\end{bmatrix}^T$$

According to grouping condition, there will be three different groups. But we can make the observation that since the constraint for c: $bw_1 + bw_2 \leq 200$ is *redundant*, we can decompose c as two unified network elements c_1 and c_2 where

$$c_1 : \textit{routingcost} = 3, \textit{bandwidth} = 200\text{Mbps}$$
$$c_2 : \textit{routingcost} = 3, \textit{bandwidth} = 200\text{Mbps}$$
$$\boldsymbol{R}_{c_1}^T = \boldsymbol{A}_{c_1} = \begin{bmatrix}1 & 0\end{bmatrix}^T, \boldsymbol{R}_{c_2}^T = \boldsymbol{A}_{c_2} = \begin{bmatrix}0 & 1\end{bmatrix}^T$$

After c is decomposed, we can invoke EQUIVAGGREGATION (Algorithm 1) and c_1 and c_2 can be aggregated with a and b respectively. We introduce the definition of *constraint redundancy* by Telgen [21] as Definition 3 and further prove that the decomposition guarantees equivalence.

Definition 3 (Redundant linear constraint – Telgen [21])**.** For a linear system whose feasible region $F = \{\boldsymbol{x} \mid \boldsymbol{A}\boldsymbol{x} \leq \boldsymbol{b}\}$, the k-the constraint $\boldsymbol{A}_k\boldsymbol{x} \leq b_k$ is redundant if and only if the feasible region $F_k = \{\boldsymbol{x} \mid \boldsymbol{A}_i\boldsymbol{x} \leq b_i, i \neq k\}$ is equal to F, i.e. $F_k = F$.

Theorem 3. For $V_i(\boldsymbol{R}_i^T, \boldsymbol{P}_i, \{\boldsymbol{A}_i^k\}, \{\boldsymbol{Q}_i^k\})$, we say V_i is redundant if and only if $\boldsymbol{A}_i^k\boldsymbol{x} \leq \boldsymbol{q}_i^k$ is redundant for all k. If and only if V_i is redundant, we can construct an equivalent network view $V' = V \setminus V_i \cup \{V_i^j\}$ where V_i is decomposed as $V_i^{(j)}(\boldsymbol{R}_i^{Tj}, \boldsymbol{P}_i^{(j)}, \emptyset, \emptyset)$ with $\boldsymbol{R}_i^T = \sum_j \boldsymbol{R}_i^{T(j)}$ and $\boldsymbol{P}_i^{(j)} = \boldsymbol{P}_i$.

Proof. **Sketch**: We still consider the criteria Equation 1a and Equation 1b and use the same symbols in Theorem 2.

First we can prove criterion Equation 1a holds whether V_i is redundant or not.

$$\begin{aligned} m_{uv} &= \bigoplus_k r_{uk} \otimes p_k^v = \bigoplus_{k \neq i} r_{uk} \otimes p_k^v + r_{ui} \otimes p_k^v \\ &= \bigoplus_{k \neq i} r_{uk} \otimes p_k^v + \left(\sum_j r_{ui}^{(j)}\right) \otimes p_k^v \\ &= \bigoplus_{k \neq i} r_{uk} \otimes p_k^v + \bigoplus r_{ui}^{(j)} \otimes p_k^{v(j)} = m'_{uv} \end{aligned}$$

For Equation 1b, first we consider the case when V_i is redundant but $V \nsim V'$. V_i is redundant so that $\forall k$, $\boldsymbol{A}_i^k\boldsymbol{x} \leq \boldsymbol{q}_i^k$ is redundant. According to Definition 3, we have the feasible regions $F^k = F_i^k = F'^k$ for all k. Since we have already proved that Equation 1a holds, according to Theorem 1, $V \sim V'$ which leads contradiction.

If V_i is not redundant but $V \sim V'$, we can similarly construct a contradiction between the definition of redundancy and the equivalency criterion.

Thus, we have proved that V_i can be equivalently decomposed if and only if V_i is redundant. □

The efficiency and privacy of equivalent decomposition depend on 1) how to identify redundant components, and 2) how to find the basis. In this paper, we use a heuristic approach which aims to simplify the selection of basis, as introduced in Algorithm 2.

Line 2 identifies the set of decomposable components V_d according to Theorem 3, i.e. $\forall v_i \in V_d$, v_i is redundant. The algorithm then decomposes these redundant components to a unit basis $\{V_j^{(i)} \mid r_{ji} \neq 0\}$ in Line 4-11. According to Theorem 3, V' after each iteration is *equivalent* to the original network view V. Finally we invoke EQUIVAGGREGATION(V') to aggregate the V_j^i with the same $\boldsymbol{R}_i^T$, which is also proved to maintain the equivalence condition as in Theorem 2. Thus, Algorithm 2 returns the equivalent network view.

For each iteration, the decomposition takes $O(n)$ time to check the condition in Line 8 and $O(ns)$ to construct the corresponding basis. The algorithm would at most have m iterations so the total execution time for decomposition is $O(mns)$ and the total execution time with EQUIVAGGREGATION is $O(mn(s+t))$.

Different algorithms exist to find the decomposable components, based on Theorem 3. For example, one can find all elements with non-redundant linear constraints, which is a well-studied problem, and get the decomposable set by calculating its inverse.

C. Privacy preservation

The equivalent aggregation and equivalent decomposition are equal to matrix factorization. While the consumer can only infer the network elements which cannot be decomposed without jeopardizing feasibility or optimality, it is impossible to infer the complete original network state without knowing the exact value of the transform matrix. Thus, Algorithm 2 can improve the privacy preservation and reduce information leak. It is noticeable that compared with some other optimization frameworks, NOVA does not require consumers to specify private information, e.g. private constraints and objective functions, which also protects the privacy of the consumers.

V. EVALUATION

In this section, we evaluate NOVA to demonstrate its efficiency and efficacy in providing accurate on-demand network views in comparison to some other abstraction models.

Algorithm 2: NOVA Equivalent Element Decomposition

Input: $V(\boldsymbol{R}, \boldsymbol{P}, \boldsymbol{A}, \boldsymbol{Q})$, F
Output: $V'(\boldsymbol{R'}, \boldsymbol{P'}, \boldsymbol{A'}, \boldsymbol{Q'})$
1 **Function** EQUIVDECOMPOSITION(V, F)
2 $V_d \leftarrow$ FINDEQUIVDECOMPOSABLE(V)
3 $V' \leftarrow V$
4 **foreach** $V_j \in V_d$ **do**
5 $S \leftarrow \emptyset$
6 **foreach** $f_i \in F$ **do**
7 $\boldsymbol{R}_i^T \leftarrow \left[\underbrace{0, \ldots, 0}_{i-1}, r_{ji}, 0, \ldots\right]$
8 **if** $r_{ij} = 1$ **then**
9 $V_j^{(i)} \leftarrow \left(\boldsymbol{R}_i^T, \boldsymbol{P_j}_i, \emptyset, \emptyset\right)$
10 $S \leftarrow S \cup \left\{V_j^{(i)}\right\}$
11 $V' \leftarrow (V' \setminus V_j) \cup \{S\}$
12 $V' \leftarrow$ EQUIVAGGREGATION(V')
13 **return** V'

A. Methodology

Performance metrics. We evaluate the performance of NOVA using the following metrics:

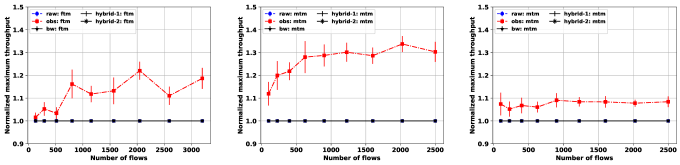
- *Optimality* and *feasibility*: To demonstrate optimality and feasibility, we generate random linear objective functions maximizing the weighted throughput [22] to demonstrate its generality and compare the results.
- *Communication overhead reduction*: We measure the communication overhead by the number of (*flow, element*) pairs contained in an network view so that smaller numbers represents better reduction.
- *Privacy preservation*: We measure the privacy preservation by the ratio of network elements in the original network view that still appear in the abstract network view. Smaller numbers represents less information leak and better privacy. As being said in Section IV-C, the non-redundant network elements cannot be reduced, we specifically evaluate the preservation of redundant elements.
- *Computation overhead*: We measure the computation overhead by the execution time of the abstraction process so that smaller numbers represents better performance.

Topology. Three topologies are used in this evaluation: Kdl (752 nodes, 1790 links), AS4755 (531 nodes, 1428 links) and AS2914 (10820 nodes, 32844 links) from two data sets: the topology zoo [23] and rocketfuel [24]. If the topology already has bandwidth information, we use it directly. Otherwise, we generate stepped values for links from edge to core. We allocate the routingcost randomly following the standard distribution around the reciprocal of bandwidth multiplied by a given constant to avoid precision issues.

Redundancy check algorithms. We use two redundancy check algorithms in our evaluation. The first, denoted as *strict redundancy check*, follows Definition 3 and can find the minimal set of non-redundant linear constraints. The second, denoted as *relaxed*, identifies redundancy by randomly selecting basis and comparing the bound with the sum of the basis and may lead to *false negative* in identifying redundancies.

Flow requests. We use two traffic patterns: *few-to-many* (*ftm*) and *many-to-many* (*mtm*). The first represents the server-client traffic pattern while the second represents the peer-to-peer traffic pattern. For each pattern, we have 9 groups with different number of flows and for each group we randomly generate 3 flow requests for each topology. The flow requests are computed with *bandwidth-only* (*bw*), *routingcost-only* (*rc*) and *hybrid* (two variants *hybrid-1* and *hybrid-2*) respectively. For *bandwidth-only* requests, we use the *strict* redundancy check. The other requests uses Algorithm 2 with different redundancy check algorithms: no check for *routingcost-only*, *strict* redundancy check for *hybrid-1* and *relaxed* redundancy check for *hybrid-2* to demonstrate the effect of how redundancy check algorithms on the performance of NOVA.

Environment and data collection. The prototype system is built with Python and uses the PuLP framework and *COIN Branch and Cut* (CBC) solver to solve linear programming.



(a) Kdl, *ftm*, normalized. (b) Kdl, *mtm*, normalized. (c) 2914, *mtm*, normalized.

Fig. 2: Normalized Maximum Weighted Throughput.

The evaluations are conducted on a laptop with Linux kernel 4.9.6, 4 quad-core Intel(R) Core(TM) i7-4700MQ @2.40GHz CPU and 16 GB memory. For each topology, we generate three different routing cost distributions and three different flow requests of the same flow size. The data are collected from the same execution.

B. Optimality and feasibility

The normalized results for using different views to solve the same random linear programming problems are compared with the original network view (*raw*) and the *one-big-switch* (*obs*). Since queries with only routing cost would not generate any linear constraints, it is omitted in this evaluation.

As demonstrated in Figure 2, we can see that NOVA *always* achieves the same optimal solution as with the original network view while the *one-big-switch* abstraction results in infeasible solutions. The reason is that *one-big-switch* abstraction does not identify the bottleneck links inside the network.

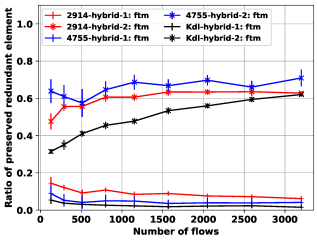
C. Privacy preservation

To demonstrate how much information consumers can learn about the original network, we use the number of preserved links to measure privacy preservation as specified in Section V-A. As demonstrated in Figure 3, we can see that despite the non-redundant links which cannot be hidden without jeopardizing equivalence, NOVA with strict redundancy checks can hide almost all the redundant links.

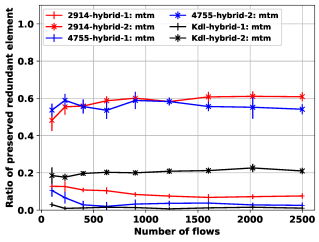
We have identified three effective factors on the privacy preservation of NOVA: 1) the redundancy check criterion, 2) flow patterns, and 3) the number of flows.

As demonstrated in Figure 3, we can see that the effect of privacy preservation is mostly determined by the redundancy check algorithm. In both traffic patterns, *hybrid-1* using strict redundancy check preserves very few links in the abstract network view (less than 10% in all three topologies) that it almost overlaps with the theoretical optimal ratio denoted by the *bandwidth-only*, while *hybrid-2* generally preserves more redundant links. The reason is that the relaxed redundancy check used in *hybrid-2* has false negative results and those redundant links are not decomposed.

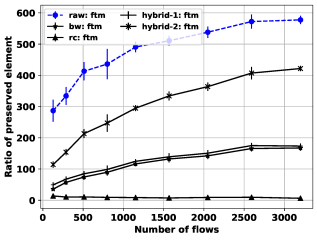
Traffic patterns have slightly less impact on the privacy preservation but we can still observe that for traffic with the *few-to-many* pattern, more links are preserved by *hybrid-2*. This is because in the *few-to-many* traffic pattern, the flows sharing a link on the "*many*" side would diverge to different paths and have no further correlation. Thus, even the link is



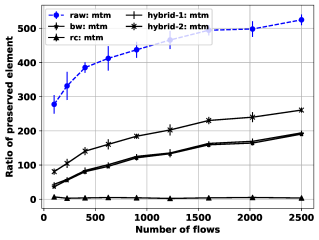
(a) *few-to-many*, normalized.



(b) *many-to-many*, normalized.

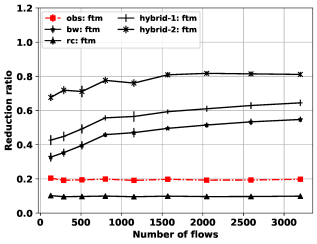


(c) Kdl, *many-to-many*, absolute.

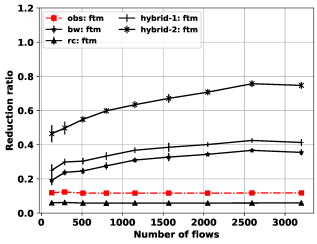


(d) Kdl, *many-to-many*, absolute.

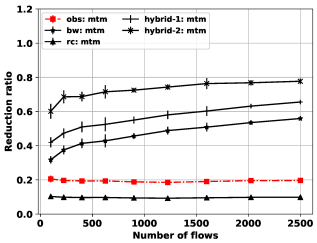
Fig. 3: Preserved links in abstract network view.



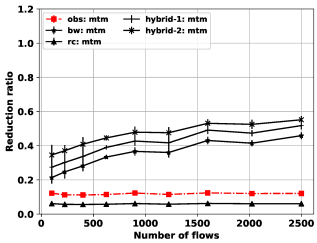
(a) 2914, *few-to-many*, normalized.



(b) Kdl, *few-to-many*, normalized.



(c) 2914, *many-to-many*, normalized.



(d) Kdl, *many-to-many*, normalized.

Fig. 4: Communication overhead reduction.

unlikely to become the bottleneck, it still cannot be identified by simple redundancy check algorithms.

The number of flows would affect the privacy preservation, as demonstrated in Figure 3c and Figure 3d. It is intuitive since more flows can generate more combinations of correlation, and reveal more information about the network. However, even with relatively large flow requests (more than 2500 flows), NOVA can protect as much as 60% of the original sliced network view in both traffic patterns using the strict redundancy check. It is also worth pointing out that when the request only contains *flow-independent* metrics, NOVA will fall back to the *end-to-end* abstraction.

D. Communication overhead reduction.

As analyzed in Section IV-A and Section IV-B, while NOVA guarantees feasibility, optimality and protects privacy, it can also reduce the communication overhead. The communication overhead is measured by the number of (*flow, element*) pairs in the network view, and the results are normalized by the value of the original network view. As demonstrated in Figure 4, we can see that depending on the effective factors, NOVA can shrink the communication overhead by a factor of 1.25 to 5.

We can see that the communication overhead reduction is affected by the same effective factors – the redundancy check algorithm, the traffic pattern and the number of flows – in a similar way as privacy preservation. The reason is that the communication overhead is mostly reduced by aggregating and decomposing redundant links. As the ratio of non-redundant links grows, the reduction is less obvious.

We can see that the theoretical lower bound (result of *bandwidth-only*) of communication overhead reduction is larger than that of privacy preservation. The reason is that most bottlenecks are usually shared by more flows, so the average number of flows on the preserved links is larger than the average in the whole slice. The gap between *hybrid-1* and *bandwidth-only* in communication overhead reduction is larger than that in privacy preservation, because flows are sent to multiple receivers from a host so links containing a single flow will not affect the privacy but still has an impact on the communication overhead.

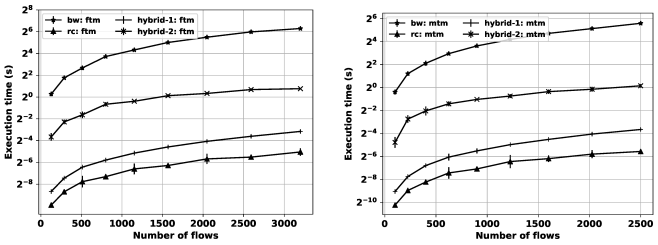
E. Computation overhead

As we can see from Figure 5, the most time-consuming part of our evaluation is to find the minimal non-redundant linear constraints, which is a well-studied problem in optimization theory and network providers can reduce this cost significantly by introducing more efficient algorithms. Even our naive solution can return an equivalent network view within 5 seconds for up to 400 flows in a large network (AS2914), which is often fast enough for many network optimization problems.

Meanwhile, the *equivalent decomposition* (represented as *hybrid-1* in Figure 5) is very efficient, which takes less than 250ms even for the largest test case with 3200 flows and a network with more than 10000 nodes (AS2914). The relaxed redundancy check algorithm takes approximately 1 second, which makes it useful in certain scenarios.

F. Summary

In this section, we evaluate the performance of NOVA thoroughly. We demonstrate that *one-big-switch* can lead to infeasible solutions while NOVA guarantees both feasibility and optimality, which enables consumers to fully utilize the network resources. Privacy is achieved by decomposing the redundant network elements. With strict redundancy check algorithms, we can reduce 60% to nearly 100% of unnecessary information leaks. Depending on the number of flow requests and traffic pattern, NOVA can improve the communication



(a) 2914, *few-to-many*. (b) 2914, *many-to-many*.

Fig. 5: Computation overhead.

time by a factor of 1.25 to 5. The computation overhead is within a reasonable range for typical uses where elephant flows usually takes tens to hundreds of seconds to finish. Thus, NOVA is useful for network providers to achieve collaborative optimization with non-administrative parties to build QoS-aware applications.

VI. Related Work

A. Demands for network views

The demands for being network-aware are quite common. For services built on top of the Internet, user experience depends heavily on the quality of networking service [25]. Previous studies [26] have already shown that obtaining end-to-end metrics can significantly improve the user experience of peer-to-peer services and content delivery networks.

Meanwhile, several studies (*e.g.*, [27]–[29]) have also addressed the need to conduct flow scheduling over the network, suggesting the importance of obtaining the correlations between different data transfers. Such demands are usually associated with traffic with large volumes, such as inter-data center communication, e.g, Google's globally-deployed B4 [15] system and global data intensive science networks [18]. Feeding these applications with more accurate network information allows them to make more intelligent operating decisions.

Another example where being aware of the network performance can be beneficial is fine-grained routing. Latest approaches such as the Software Defined Internet Exchange point (SDX) [30] have enabled Autonomous Systems to set up fine-grained forwarding rules. With the ability to query the expected network performance, an AS would be able to make routing decisions based not only on the cost, but also on the real-time quality of service. Meanwhile, such information can also be provided to QoS-based routing protocols [19], [20].

SOL [31] and CoFlow [27] are SDN-based network optimization frameworks which provide abstractions to simplify the modeling of network optimization problems. However, it would require the optimizer to provide all the information to the network, which jeopardizes the privacy. General collaborative optimization [32]–[34] typically protects the privacy by multiplying a monomial matrix. NOVA enables collaborative optimization by providing the network views to the optimizer, while conducting equivalent transformations to reduce the communication overhead as well as protect the privacy.

B. Providing network view

The most straight-forward way of providing network views is to use its graph representation. Several routing protocols [8]–[11] including OSPF and IS-IS conceptually provide such an abstraction of the network and it is also adopted by the I2RS (Interface to Routing System) IETF Working group [35]. Modern SDN controllers [1]–[4] also provide the global view using the annotated graph model.

The hose model [6] is first introduced for VPN provisioning in 1999. Each endpoint is associated with a *hose* in this model and the details of the actual VPN tunnels are hidden. It is sometimes referred to as the *one-big-switch* in the context of SDN because the network is abstracted as a single logical switch in this model. Because of its simplicity, the hose model is widely used, for example, by many network programming languages [36], [37]. SDX also uses this model to encapsulate the underlying network topology. Data center fabrics are highly customized for scalability [7] and can be modeled as a non-blocking switch where congestion only occurs on access links [14], thus, the *one-big-switch* abstraction is also widely used for data center flow scheduling and tenant resource provisioning [27]–[29].

The pipe model is mostly used by web-based applications or measurement frameworks, which have no control over the network. The pipe model consists of several flows (host pairs) and provides a single pipe for each flow (pair) with the associated metrics. PerfSONAR [38], Meridian[39] and ClosestNode [40] are some concrete examples which provide such end-to-end network views based on measurement, while P4P [26] and the ALTO (Application-Layer Traffic Optimization) protocol [12] are leveraging the network providers' information.

NOVA is similar to ALTO in the sense that in both cases information is provided by the network to non-administrative consumers, which is likely to achieve better accuracy. Meanwhile, we overcome the limitations of ALTO by adopting the equivalence abstraction to provide fine-grained metrics, in particular the *flow correlations*, which makes it possible to suffice the demands from a broader range of applications. This underlying philosophy also distinguishes NOVA from other (especially QoS related) routing protocols and network views based on topological aggregation [10].

VII. Conclusion

In this paper, we systematically study the problem of providing an accurate on-demand network view which is general enough to suffice the requirement of heterogeneous optimization problems. Our abstraction is based on the principle of *equivalence* which guarantees generality, feasibility and optimality. We design the NOVA framework to construct equivalent network views with enhanced privacy preservation and evaluate its performance compared with some well-known network view abstractions. While we have established the theoretical foundations and guidelines of constructing on-demand network optimization view abstractions, essential functionalities such as communication protocols, easy-to-use API design and commercial models are still not fully discovered. We plan

to explore these missing functionalities in the future, along with the implementation and deployment in real networks.

VIII. Acknowledgment

The authors would like to thank Chen Gu, Jingxuan Zhang, Shenshen Chen, Xiao Lin, Haoran Wang and Haizhou Du for their help during the preparation of the paper. We would also like to thank the reviewers for their valuable feedback.

This research is sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

This work is also supported by the *National Science Foundation* (CC-IIE 1440745) and the *National Natural Science Foundation of China* (No. 61472213 and No. 61502267).

References

[1] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: towards an operating system for networks," vol. 38, no. 3, pp. 105–110.

[2] T. Koponen, M. Casado, N. Gude, J. Stribling, L. Poutievski, M. Zhu, R. Ramanathan, Y. Iwata, H. Inoue, T. Hama, and others, "Onix: A distributed control platform for large-scale production networks." in *OSDI*, vol. 10, pp. 1–6.

[3] P. Berde, M. Gerola, J. Hart, Y. Higuchi, M. Kobayashi, T. Koide, B. Lantz, B. O'Connor, P. Radoslavov, W. Snow, and G. Parulkar, "ONOS: Towards an open, distributed SDN OS," in *Proceedings of the Third Workshop on Hot Topics in Software Defined Networking*, ser. HotSDN '14, pp. 1–6.

[4] J. Medved, R. Varga, A. Tkacik, and K. Gray, "Opendaylight: Towards a model-driven SDN controller architecture," in *Proceeding of IEEE International Symposium on a World of Wireless, Mobile and Multimedia Networks 2014*.

[5] Y. Rekhter, T. Li, and S. Hares, "A border gateway protocol 4 (BGP-4)."

[6] P. Mishra, K. K. Ramakrishnan, and J. E. van der Merwe, "A flexible model for resource management in virtual private networks," in *Proc. of ACM SIGCOMM*.

[7] A. Greenberg, J. R. Hamilton, N. Jain, S. Kandula, C. Kim, P. Lahiri, D. A. Maltz, P. Patel, and S. Sengupta, "VL2: a scalable and flexible data center network," in *ACM SIGCOMM computer communication review*, vol. 39, pp. 51–62.

[8] J. Moy, "OSPF version 2."

[9] D. Oran, "OSI IS-IS intra-domain routing protocol."

[10] T. Vu, A. Baid, H. Nguyen, and D. Raychaudhuri, "EIR: Edge-aware interdomain routing protocol for the future mobile internet."

[11] W. C. Lee, "Topology aggregation for hierarchical routing in ATM networks," vol. 25, no. 2, pp. 82–92.

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

[13] S. Uludag, K.-S. Lui, K. Nahrstedt, and G. Brewster, "Analysis of topology aggregation techniques for QoS routing," vol. 39, no. 3, p. 7.

[14] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. O. Guedes, "Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks." in *WIOV*.

[15] S. Jain, A. Kumar, S. Mandal, J. Ong, L. Poutievski, A. Singh, S. Venkata, J. Wanderer, J. Zhou, M. Zhu, and others, "B4: Experience with a globally-deployed software defined WAN," in *Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM*, pp. 3–14.

[16] A. Kumar, S. Jain, U. Naik, A. Raghuraman, N. Kasinadhuni, E. C. Zermeno, C. S. Gunn, J. Ai, B. Carlin, M. Amarandei-Stavila, and others, "BwE: Flexible, hierarchical bandwidth allocation for WAN distributed computing," in *ACM SIGCOMM Computer Communication Review*, vol. 45, pp. 1–14.

[17] H. Xu and B. Li, "Joint request mapping and response routing for geo-distributed cloud services," in *INFOCOM, 2013 Proceedings IEEE*, pp. 854–862.

[18] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski, "The science DMZ: A network design pattern for data-intensive science," vol. 22, no. 2, pp. 173–185.

[19] J. L. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop routing in the internet," vol. 10, no. 4, pp. 541–550.

[20] H. Geng, X. Shi, X. Yin, Z. Wang, and H. Zhang, "Algebra and algorithms for efficient and correct multipath QoS routing in link state networks," in *Quality of Service (IWQoS), 2015 IEEE 23rd International Symposium on*, pp. 261–266.

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

[22] Y. Bartal, F. Y. Chin, M. Chrobak, S. P. Fung, W. Jawor, R. Lavi, J. Sgall, and T. Tich, "Online competitive algorithms for maximizing weighted throughput of unit jobs," in *Annual Symposium on Theoretical Aspects of Computer Science*, pp. 187–198.

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

[24] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with rocketfuel," vol. 12, no. 1, pp. 2–16.

[25] R. K. Mok, E. W. Chan, and R. K. Chang, "Measuring the quality of experience of HTTP video streaming," in *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*, pp. 485–492.

[26] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4p: Provider portal for applications," vol. 38, no. 4, pp. 351–362.

[27] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *Proceedings of the 11th ACM Workshop on Hot Topics in Networks*, pp. 31–36.

[28] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks." in *NSDI*, vol. 10, pp. 19–19.

[29] M. Alizadeh, S. Yang, M. Sharif, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "pfabric: Minimal near-optimal datacenter transport," in *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 435–446.

[30] A. Gupta, L. Vanbever, M. Shahbaz, S. P. Donovan, B. Schlinker, N. Feamster, J. Rexford, S. Shenker, R. Clark, and E. Katz-Bassett, "SDX: A software defined internet exchange," vol. 44, no. 4, pp. 551–562.

[31] V. Heorhiadi, M. K. Reiter, and V. Sekar, "Simplifying software-defined network optimization using SOL," in *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*, pp. 223–237.

[32] Y. Hong, "Privacy-preserving collaborative optimization."

[33] J. Vaidya, "Privacy-preserving linear programming," in *Proceedings of the 2009 ACM symposium on Applied Computing*, pp. 2002–2007.

[34] J. Li and M. J. Atallah, "Secure and private collaborative linear programming," in *Collaborative Computing: Networking, Applications and Worksharing, 2006. CollaborateCom 2006. International Conference on*, pp. 1–8.

[35] J. Medved, N. Bahadur, H. Ananthakrishnan, X. Liu, R. Varga, and A. Clemm, "A data model for network topologies."

[36] C. Monsanto, J. Reich, N. Foster, J. Rexford, D. Walker, and others, "Composing software defined networks." in *NSDI*, pp. 1–13.

[37] A. Voellmy, J. Wang, Y. R. Yang, B. Ford, and P. Hudak, "Maple: simplifying SDN programming using algorithmic policies," in *ACM SIGCOMM Computer Communication Review*, vol. 43, pp. 87–98.

[38] A. Hanemann, J. W. Boote, E. L. Boyd, J. Durand, L. Kudarimoti, R. Lapacz, D. M. Swany, S. Trocha, and J. Zurawski, "PerfSONAR: A service oriented architecture for multi-domain network monitoring," in *International Conference on Service-Oriented Computing*, pp. 241–254.

[39] B. Wong, A. Slivkins, and E. G. Sirer, "Meridian: A lightweight network location service without virtual coordinates," in *ACM SIGCOMM Computer Communication Review*, vol. 35, pp. 85–96.

[40] B. Wong and E. G. Sirer, "ClosestNode.com: an open access, scalable, shared geocast service for distributed systems," vol. 40, no. 1, pp. 62–64.