An Objective-Driven On-Demand Network Abstraction for Adaptive Applications

Kai Gao, *Member, IEEE*, Qiao Xiang, *Member, IEEE*, Xin Wang, Yang Richard Yang, *Member, IEEE*, and Jun Bi, *Senior Member, IEEE*

***Abstract*— Revealing an abstract view of the network is essential for the new paradigm of developing network-aware adaptive applications that can fully leverage the available computation and storage resources and achieve better business values. In this paper, we introduce ONV, a novel abstraction of flow-based on-demand network view. The ONV models network views as linear constraints on network-related variables in application-layer objective functions, and provides "equivalent" network views that allow applications to achieve the same optimal objectives as if they have the global information. We prove the lower bound for the number of links contained in an equivalent network view, and propose two algorithms to effectively calculate on-demand equivalent network views. We evaluate the efficacy and the efficiency of our algorithms extensively with real-world topologies. Evaluations demonstrate that the ONV can simplify the network up to 80% while maintaining an equivalent view of the network. Even for a large network with more than 25 000 links and a request containing 3000 flows, the result can be effectively computed in less than 1 min on a commodity server.**

***Index Terms*— Software-defined networking, routing algebra, quality of service, resource abstraction.**

I. Introduction

LARGE-SCALE distributed systems, such as geo-distributed data centers [1] and international scientific research programs [2], [3], have components (data centers, sites, etc.) located in different cities, countries and even continents. To ensure connectivity and minimal performance guarantees, these components are usually connected by tunnels with resource reservations.

Advanced network management technologies such as Software Defined Networking (SDN) have enabled network service providers to provide on-demand resource reservations, such as AT&T's Domain 2.0 [4] and ESNet's OSCAR system [5]. Network tenants can adjust the reservations flexibly to better match their demands.

However, demands of large-scale distributed systems are usually not fixed because of data replications and service load balancing – the same transfer job can be done using different components. In the meantime, orchestration systems such as Microsoft's Clarinet system [6] have been developed to optimize the large-scale query jobs across different geo-distributed data centers based on real-time inter-connection qualities, *i.e.*, *the optimal demands of such systems depend on the available resources*.

It is quite common that tenants decide their optimal demands based on *a certain objective function of available resources and end-to-end metrics*, such as high tunnel utilization [1], flow completion time [7], job completion time [6], throughput [8], etc. Without the ability to accurately know the available resources, a tenant can only make blind guesses which may lead to conservative or unrealistic reservations, and hurt the tenant's quality of service. Thus, it's becoming increasingly important that network service providers offer *on-demand resource abstractions* to help tenants better exploit the flexibility of *on-demand resource reservations*.

SDN enables a network to collect information from all the devices and construct a global view, which may contain essential quality of service (QoS) metrics such as available bandwidth, loss rate and routing cost values, which are critical to performance of distributed applications.

Unfortunately, while northbound APIs for "apps" (management programs) to access the global view have been provided by many SDN controllers (*e.g.*, [9]–[12]), they are usually not open to non-administrative parties. Major concerns include privacy and security, because the global view contains sensitive information that can be leveraged to conduct attacks on the SDN infrastructure [13]. Also, the global view is not friendly to program with because it can contain a lot of redundant information and lead to unnecessary communication overhead.

Thus, a problem arises on how to provide an abstract network view which can both eliminate these drawbacks and

Manuscript received November 4, 2017; revised June 9, 2018; accepted January 25, 2019; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Mascolo. Date of publication March 26, 2019; date of current version April 16, 2019. This work was supported in part by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Grant W911NF-16-3-0001, in part by the National Science Foundation under Grant CC-IIE 1440745, in part by the National Natural Science Foundation of China under Grant 61472213 and Grant 61502267, and in part by the National Key Research and Development Plan of China under Grant 2017YFB0801701. *(Corresponding author: Kai Gao.)*

K. Gao was with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China, and also with the Department of Computer Science, Yale University, New Haven, CT 06511 USA. He is now with the College of Cybersecurity, Sichuan University, Chengdu 610065, China (e-mail: kaigao@scu.edu.cn).

Q. Xiang and Y. R. Yang are with the Department of Computer Science, Yale University, New Haven, CT 06511 USA (e-mail: qiao.xiang@yale.edu; yry@cs.yale.edu).

X. Wang is with the Department of Computer Science and Technology, Tongji University, Shanghai 201804, China, and also with the Key Laboratory of Embedded System and Service Computing, Ministry of Education, Beijing 100816, China (e-mail: 13xinwang@tongji.edu.cn).

J. Bi is with the Institute for Network Sciences and Cyberspace, Tsinghua University, Beijing 100084, China, and also with the Beijing National Research Center for Information Science and Technology, Tsinghua University, Beijing 100084, China (e-mail: junbi@tsinghua.edu.cn).

Digital Object Identifier 10.1109/TNET.2019.2899905

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 reservations are either rejected, or lead to over-subscribed tunnels which may eventually lead to congestion.
- *Generality*: The abstract view 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 abstract view must be able to protect the privacy of the network service provider, making it difficult for malicious applications to infer the global information.
- *Efficiency*: The abstract view should not introduce too much computation/communication overhead, even with moderately large networks and workloads.

Existing abstractions [14]–[23] usually target at a certain type of scenario and cannot support applications which require fine-grained QoS metrics. For example, many of these abstractions cannot 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 [1], [24], [25] and scientific computing platforms [26].

In this paper, we take the first step towards providing high-quality network information for network-aware adaptive applications with ONV, an abstraction for flow-based On-demand Network View. Based on the observation that network information is eventually used by applications to conduct optimizations, ONV provides *equivalent* network view which satisfies the aforementioned properties simultaneously.

The main contributions in this paper include:

- We systematically investigate the problem of providing on-demand network view for network-aware adaptive applications with heterogeneous optimization goals, a missing functionality from current SDN northbound API design.
- We address the challenges by proposing ONV, an abstraction for flow-based on-demand network view. ONV is based on the concept of *equivalent network view*. We derive the criteria of equivalent network view and gives the lower bound of number of links.
- We propose two algorithms which conduct equivalent network transformations to obtain the equivalent network view.
- We implement a prototype of ONV and evaluate its performance using real applications over simulated networks of real topologies. Evaluations show that ONV 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 (real ISP network topologies with up 10000 nodes and 30000 links) and large workloads (more than 3000 flows).

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. Formal descriptions of the abstraction problem and the *equivalent network view* are then given in Section III and Section IV respectively, followed by the transformation algorithms in Section V. We evaluate the prototype and analyze the results in Section VI. Finally, we discuss the related work and give conclusions in Section VII and Section VIII respectively.

II. Motivation

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

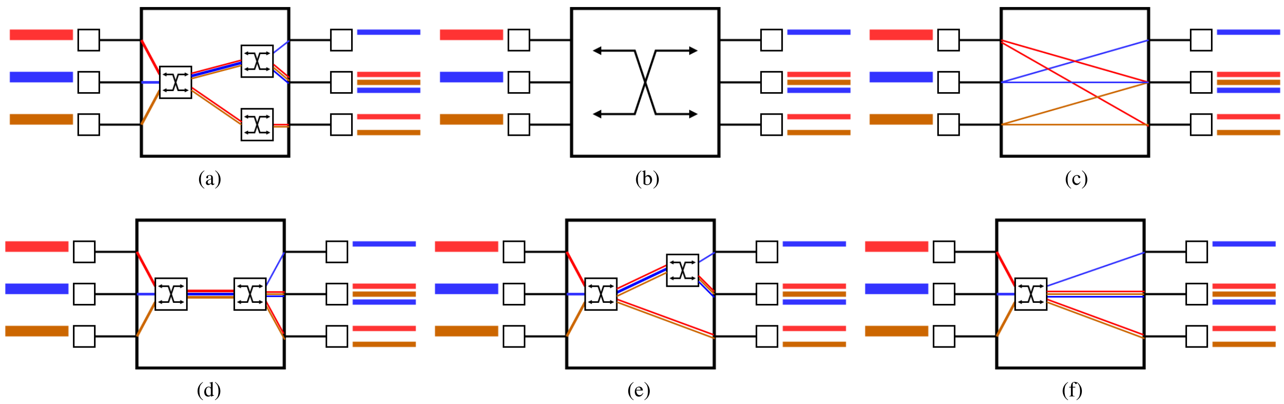
The naive approach returns the slice containing 1) all the links on the flow paths with the associated bandwidth information, and 2) how the links are shared by the flows, as shown in Fig. 1(a) in this case. However, this can lead to information leaks so that malicious applications may leverage this service to infer the network information, which jeopardizes the privacy of the network service 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 Fig. 1(b). However, the application would only know the available bandwidth on the ingress/egress "port". If the bottleneck is the upper middle link, it is not propagated to the application. Thus, the application 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 reduce the throughput of all the flows sharing the same link, leading to an overall performance degradation.

The end-to-end abstraction as demonstrated in Fig. 1(c) is completely 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 application.

Topology aggregation [22] is a common technique to reduce 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 Fig. 1(d), 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 Fig. 1(e). One may think we can just delete one of them. However, if the application asks for the



The application has 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. (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.

end-to-end routing metrics such as hop count at the same time, deletion would return incorrect values for the two flows.

Meanwhile, if we already know that the upper middle link would not be the bottleneck, the network view can be further reduced, as demonstrated in Fig. 1(f). 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 ONV with efficient algorithms to compute the equivalent network view.

III. On-Demand Network Views

In this section, we formally define the model of *on-demand network views* and the theoretical foundations – the variant routing metric algebra, and the unified network element. We also discuss how to encode on-demand network views.

A. Basic Settings

We first formally define the models for the networks and applications discussed in this paper, as summarized in Table I.

1) Network: A network is a graph of arbitrary topology consisting of a set of M unified network elements or simply *element* (defined in Section III-C). For each element, the network can provide two kinds of routing metrics:

- A *flow-independent* metric represents a metric whose value is *independent* of the flow correlations, *i.e.*, 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 [27] include hop count, delay, and loss rate.[1] We also require these metrics to be *linearly addictive*, *i.e.*, for a given metric w and two network elements a and b, $w(a+b) = w(a) \oplus w(b)$.

[1]Delay, 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*, as used in network tomography [28].

TABLE I

SYMBOLS FOR NETWORK VIEW ABSTRACTION

Scope	Symbol	Meaning
Network	M	Number of network elements
	$\mathcal{E}$	Set of network elements, $\mathcal{E} = \{e_1, \ldots, e_M\}$
	K_i	Number of flow-independent metrics
	K_c	Number of flow-correlated metrics
	$p_{j,k}$	The k-th flow-independent metric of e_j
	P	The matrix of flow-independent metrics
	$q_{j,k}$	The k-th flow-correlated metric of e_j
	Q	The matrix of flow-correlated metrics
Application	N	The number of flows (potential tunnels)
	$\mathcal{F}$	The set of flows $\mathcal{F} = \{f_1, \ldots, f_N\}$
	$x_{i,k}$	The k-th end-to-end metric for f_i
	X	The matrix of end-to-end metric
	$y_{i,k}$	f_i's allocation of the k-th resource
	Y	The matrix of resource allocations
	K_p	Number of private variables
	z_k	The k-th private variable
	Z	The 1-row matrix of private variables
	U	The objective function
	K_{pc}	Number of private constraints
	g_k	The k-th private constraint function
Routing	$a_{i,j}$	Whether element j appears in flow i's path
	A	Routing matrix
Routing algebra	$\mathcal{P}$	Set of paths
	$\mathcal{S}$	Set of metrics
Network view	V	Network view represented by a triplet
	$V_j(\boldsymbol{v_j})$	The j-th component (and index vector) of V
	T	The view-abstraction transformation
Common	mat^i	The i-th column of matrix *mat*
	mat_i	The i-th row of matrix *mat*

- A *flow-correlated* metric represents a network resource that is 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. We require the resource constraints to be linear, *i.e.*, for a given resource and two flows f_1 and f_2, $w(\{f_1\}) + w(\{f_2\}) = w(\{f_1, f_2\})$.

Without loss of generality, we number the elements from 1 to M where the j-th element is denoted as e_j. Let $\mathcal{E} = \{e_1, \ldots, e_M\}$. Assume each element has K_i flow-independent metrics and K_c flow-correlated metrics.

TABLE II
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

For the j-th element e_j, the k-th flow-independent metric is denoted as $p_{j,k}$ and the k-th flow-correlated metric is denoted as $q_{j,k}$. Let $P = (p_{j,k})_{M \times K_i}$ and $Q = (q_{j,k})_{M \times K_c}$.

2) Application: Each application contains a set of N unidirectional flows, which is based on the observation that applications may have asymmetric traffic demands. An application has a *private* objective function, which depends on three types of information as listed below and is subject to some private constraints:

- Per-flow end-to-end metrics: An end-to-end metric represents the end-to-end performance a certain flow can obtain, such as delay, loss rate, etc. End-to-end metrics correspond to the flow-independent metrics.
- Per-flow resource allocations: A per-flow resource allocation represents how much resource (e.g., bandwidth) a given flow can use, and corresponds to a flow-correlated metric.
- Private variables: A private variable represents information that is not known by the network provider, such as CPU utilization, available memory or even bandwidth constraints in a private network.

The flows are numbered from 1 to N and the i-th flow is denoted as f_i. Let $\mathcal{F} = \{f_1, \ldots, f_N\}$. Assume K_i represents the number of end-to-end metrics, K_c represents the number of resources, and K_p represents the number of private variables. For the i-th flow, we use $x_{i,k}$ to denote its k-th end-to-end metric and $y_{i,k}$ to denote its allocation for the k-th resource. We number the private variables from 1 to K_p and denote the k-th variable as z_k.

We use $U(X, Y, Z)$ to represent the private objective function, where $X = (x_{i,k})_{N \times K_i}$, $Y = (y_{i,k})_{N \times K_c}$, and $Z = (z_k)_{1 \times K_p}$. Without loss of generality, we assume smaller objective values indicate better results. For objective functions that an application wants to maximize, we can easily construct $U'(X, Y, Z) = -U(X, Y, Z)$ and use U' as the objective function instead.

Assume there are K_{pc} private constraints, and the i-th private constraint is denoted as $g_i(X, Y, Z) \leq 0$. Both U[2] and g_i are arbitrary functions with the mild assumption that the application can find the minimum objective under all private constraints and additional linear constraints on X and Y.

3) Routing: We consider the case where the tunnels are simple paths, *i.e.*, there are no loops or branches. Let $a_{i,j}$ denote whether f_i traverses e_j and routing matrix $A = (a_{i,j})_{N \times M}$.

[2]Even though the correctness of this paper does not have any additional assumption on U as long as the application can solve it under linear constraints and the given set of private constraints, a common assumption in practice is that the objective function is concave.

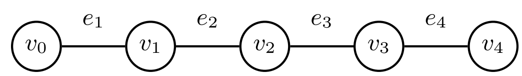


Fig. 2. Topology to demonstrate different definitions of path.

B. The Variant Routing Metric Algebra

The routing metric algebra, introduced by Sobrinho to implement QoS-based routing [29], is based on *path concatenation*. It can be represented as $(\mathcal{P}, \mathcal{S}, w, \circ, \oplus, \preceq)$. $\mathcal{P}$ is the set of paths and $\mathcal{S}$ is a closed set of metrics. The weight function w maps a path from $\mathcal{P}$ to a given metric in $\mathcal{S}$. The concatenation operator $\circ$ is a binary operator on $\mathcal{P}$, which takes two paths and returns a new one. Operator $\oplus$ is a binary operator on $\mathcal{S}$ and $\preceq$ is a binary relation on $\mathcal{S}$.

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 $\otimes : \mathbb{N} \times \mathcal{S} \mapsto \mathcal{S}$ operator to linearize the metric calculation. The operator basically means the same link is traversed for multiple times.

Second, we relax the constraint on path concatenation in the original algebra, by extending the meaning of path from a walk of nodes to a set of *unified network elements* (defined in Section III-C). Consider the network in Fig. 2, the only valid path from v_0 to v_4 in the original QoS algebra is $\langle v_0, v_1, v_2, v_3, v_4 \rangle$, but with our algebra a path can be *any* permutation of $\{e_1, e_2, e_3, e_4\}$ (24 combinations).

We use $(\mathcal{P}, \mathcal{S}, w, \circ, \oplus, \preceq, \otimes)$ to describe our variant routing algebra. $(\mathcal{S}, \oplus)$ is a semigroup so that $\oplus$ is *commutative* and *associative*. We also require that the $\otimes$ operator is *distributive* over $\oplus$. Concrete examples of some common routing metrics are demonstrated in Table II.

C. 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 a constraint on the total throughput passing through this DPI node. From the applications' perspective, it is not different from a bottleneck link.

However, certain metrics may only appear on certain types of network elements. 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 II.

Links, nodes and middleboxes are transformed into unified elements differently. For example, a duplex link should be treated as two unified network elements because flows could traverse it from both directions. Meanwhile, a middlebox should be treated as a single element because flows from different directions are throttled by the computation capability – a single bottleneck.

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 metrics' semantics rather than how the metrics are computed/constrained.

D. Abstract Network View

We use the symbols defined in Section III-A and formally define the on-demand network view.

Flow-independent metrics are formulated as *equations* according to the variant routing metric algebra. For example, the hop count between two end hosts is equal to the sum of hop counts of each link on the path. We have:

$$x_{i,k} = \bigoplus_j a_{i,j} \otimes p_{j,k} \Leftrightarrow X = A \times P.$$

If f_i consumes the same resource on all network elements it traverses, the *flow-correlated* metrics are formulated as *linear constraints*. The total resource consumption of all the flows on a single element must not exceed the available amount, i.e.,

$$\sum_i a_{i,j} y_{i,k} \le q_{j,k} \Leftrightarrow A^{\mathrm{T}} Y \le Q.$$

Thus, the network view can be formulated as a tuple of three elements: $V = (A, P, Q)$. Based on this network view model, an *abstraction* can be defined as follows:

Definition 1 (Network View Abstraction): A network view abstraction is a transform function T which takes the original network view V and returns an *abstract* view V', i.e.,

$$V'(A, P, Q) = T\left(V(A, P, Q)\right)$$

E. Encoding Abstract Network Views

We use *flow path map* and *element map* to efficiently encode an abstract network view. The flow path map, as the name suggests, is a dictionary object which maps a flow identifier to its flow path. Each flow path is a list of *element identifiers*. Each element identifier uniquely represents a unified network element, and each appearance in a flow path indicates that the flow traverses the corresponding network element once. The element map is a dictionary object which maps an element identifier to its properties. Properties are encoded in a `key: value` style. The Backus-Naur Form (BNF) for our abstract network view encoding is given in Fig. 3. An example is given in Fig. 4, which uses the topology in Fig. 2 with one end-to-end metric ("routingcost") and one flow-correlated metric ("bandwidth"). There are two flows, one from v_0 to v_4 and the other from v_4 to v_0. Each link is denoted as two elements, where $e_i = (v_{i-1}, v_i)$ for $i \in [1, 4]$ and $e_i = (v_{i-4}, v_{i-5})$ for $i \in [5, 8]$. However, this abstract network view is not optimal because it contains some redundant information.

$\langle$AbstractNetwokView$\rangle \models \{$ $\langle$FlowPathMap$\rangle$, $\langle$ElementMap$\rangle$ $\}$
$\langle$FlowPathMap$\rangle \models$"flow-path-map" : $\{\langle$FlowPathEntries$\rangle\}$
$\langle$FlowPathEntries$\rangle \models \langle$FlowPathEntry$\rangle$ | $\langle$FlowPathEntry$\rangle$, $\langle$FlowPathEntries$\rangle$
$\langle$FlowPathEntry$\rangle \models$ *FlowId* : $\langle$FlowPath$\rangle$
$\langle$FlowPath$\rangle \models$ *ElementId* | *ElementId* , $\langle$FlowPath$\rangle$
$\langle$ElementMap$\rangle \models$"element-map" : $\{$ $\langle$ElementEntries$\rangle$ $\}$
$\langle$ElementEntries$\rangle \models \langle$ElementEntry$\rangle$ | $\langle$ElementEntry$\rangle$, $\langle$ElementEntries$\rangle$
$\langle$ElementEntry$\rangle \models$ *ElementId* : $\langle$ElementAttributes$\rangle$
$\langle$ElementAttributes$\rangle \models \langle$ElementAttribute$\rangle$ | $\langle$ElementAttribute$\rangle$, $\langle$ElementAttributes$\rangle$
$\langle$ElementAttribute$\rangle \models$ *AttributeName* : *AttributeValue*

Fig. 3. The BNF for abstract network view encoding.

```
{
  "flow-path-map": {
    "f1": ["e1", "e2", "e3", "e4"],
    "f2": ["e5", "e6", "e7", "e8"]
  },
  "element-map": {
    "e1": { "routingcost": 1, "bandwidth": "100Mbps" },
    "e2": { "routingcost": 1, "bandwidth": "100Mbps" },
    "e3": { "routingcost": 1, "bandwidth": "50Mbps"  },
    "e4": { "routingcost": 1, "bandwidth": "100Mbps" },
    "e5": { "routingcost": 1, "bandwidth": "100Mbps" },
    "e6": { "routingcost": 1, "bandwidth": "100Mbps" },
    "e7": { "routingcost": 1, "bandwidth": "50Mbps"  },
    "e8": { "routingcost": 1, "bandwidth": "100Mbps" }
  }
}
```

Fig. 4. Encoding an abstract network view for Fig. 2.

Since the number of flows is fixed, the size of an abstract network view is mostly determined by the number of elements and their appearances in the flow path map. We use $\|V\|$ to denote the size of a view, which is measured by the number of unified network elements.

IV. Equivalent Network View

In this section, we introduce *equivalent network view* and an effective criterion to verify the equivalence. Furthermore, we give a lower bound of the number of unified network elements contained in an equivalent on-demand network view.

A. Equivalence of On-Demand Network Views

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

Consider an application as we model in Section III-A, the optimization problem can be formulated as

$$\begin{aligned} U_{\min} = \ & \min U(X, Y, Z) \\ & \textit{s.t}\ X = A \times P, \\ & \quad O \le A^{\mathrm{T}} Y \le Q, \\ & \quad g_i(X, Y, Z) \le 0, \quad \forall i \in [1, K_{pc}]. \end{aligned}$$

Let $U_{\min}(V)$ denote the optimal objective for a given network view V. We define "equivalence" as follows:

Definition 2 (Network View Equivalence): Two network views V and V' are *equivalent*, if and only if for any application with flows $\mathcal{F}$ and objective function U, $U_{\min}(V) = U_{\min}(V')$.

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 $V(A, P, Q)$ and $V'(A', P', Q')$ are *equivalent* if and only if for any application with flows $\mathcal{F}$

$$A \times P = A' \times P' \tag{1a}$$

$$R = \left\{Y \mid A^{\mathrm{T}}Y \leq Q\right\} = \left\{Y \mid {A'}^{\mathrm{T}}Y \leq Q'\right\} = R' \tag{1b}$$

Proof: For two network views V and V', we use $V \sim^* V'$ and $V \sim V'$ to represent that V and V' are equivalent by the criterion and are equivalent by definition respectively. It can be easily proved that if $V \sim^* V'$, $V \sim V'$ because they have exactly the same domain space and thus the same image space. For the other direction, we prove it by contradiction.

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

Assume Equation 1a does not hold. For any $X = A \times P \neq A' \times P' = X'$, we find one entry that is not equal in X and X', say $x_{i,k} \neq x'_{i,k}$ and construct an objective function $U(X, Y, Z) = x_{i,k}$ or $U(X, Y, Z) = -x_{i,k}$ based on whether smaller $x_{i,k}$ is better. Thus, the two network views have different optimal values and are not equivalent by definition, which contradicts with our assumption.

Assume Equation 1b does not hold, $\exists \hat{Y} \in (R \setminus R') \cup (R' \setminus R)$. Without loss of generality, let $\hat{Y} \in R \setminus R'$. Since $\hat{Y} \notin R'$, there exist j, k such that $({A'}^{\mathrm{T}}\hat{Y})_{j,k} = \hat{u} > q_{j,k}$. Now we construct a linear objective function $U(X, Y, Z) = -({A'}^{\mathrm{T}}Y)_{j,k}$, and assume the optimal objectives are u and u' respectively. We have $u \leq -\hat{u} < -q_{j,k} = u'$ which means the objective function has different optimal objective values for the two network views. Again, we get a contradiction.

Thus, we can conclude that if $V \sim V'$, $V \sim^* V'$ and the criterion is both sufficient and necessary. $\square$

We have proved that the network views satisfying this criterion also satisfy Definition 2. Thus, it allows us to effectively verify two network views are equivalent for applications with arbitrary objective functions and arbitrary fine-grained routing metrics as long as they fit in the *variant routing metric algebra*.

B. Lower Bound of $\|V\|$

For a given on-demand network view, there exists a set of equivalent network views which construct an equivalent class. To improve privacy and reduce communication overhead, one might want to return the *minimal* equivalent network view.

In this section, we present two extreme cases:

1) only flow-independent information is requested, and
2) only flow-correlated information is requested.

As we demonstrate in Section V-B, the general case can be processed in two stages and each stage corresponds to an extreme case.

1) Lower Bound of $\|V\|$ for Flow-Independent Metrics: When only flow-independent information is requested, the equivalent network view only needs to provide the same X for each flow. Since $\|V\|$ equals the column rank of A, the equivalent network view with the minimal $\|V\|$ has the smallest column rank of A. The problem is equivalent to the *optimal non-negative matrix factorization problem*, which has been proved to be NP-Hard [30].

2) Lower Bound of $\|V\|$ for Flow-Correlated Information: When only flow-correlated information is requested, equivalent network views should return the *same feasible region*, which is determined by a set of linear constraints. Since each constraint is essentially one network element, $\|V\|$ is directly related to the number of constraints.

We first introduce the definition of redundant linear constraint by Telgen [31] and propose Theorem 2.

Definition 3 (Redundant Linear Constraint – elgen [31]): For a linear system whose feasible region $R = \left\{\vec{x} \mid A\vec{x} \leq \vec{b}\right\}$, the k-the constraint $A_k\vec{x} \leq b_k$ is redundant if and only if the feasible region $R_k = \{\vec{x} \mid A_i\vec{x} \leq b_i, i \neq k\}$ is equal to R, i.e. $R_k = R$.

Theorem 2: If only flow-correlated information is requested, $\|V\|$ is minimal if and only if the corresponding constraint set $C = \{c_j | c_j : A_j^{\mathrm{T}}Y \leq Q_j\}$ has no redundant constraints.

Proof: $\Rightarrow$: Consider the opposite that $\|V\|$ is minimal but C contains redundant constraints. According to the definition of *redundant constraints* by Telgen [31], we can remove the redundant constraints but still obtain the same feasible region. Thus, we have a network view with a smaller $\|V\|$ and it leads to a contradiction.

$\Leftarrow$: Consider the opposite that C contains no redundant constraints but $\|V\|$ is not minimal. Let C' represents the equivalent constraint set of the minimal size and $\|C'\| < \|C\|$. Since C and C' have the same feasible region, they also have the same feasible region as $C \cup C'$. Since C contains no redundant constraints, there exists at least one $c^* \in (C \cup C') \setminus C$ which is redundant. Thus, C' contains a redundant constraint and cannot have the minimal number of element, which leads to a contradiction with our assumption. $\square$

The problem of finding redundant linear constraints has been widely studied. For example, Paulraj and Sumathi [32] have summarized several algorithms to find the redundant constraints. Since there exists polynomial time algorithms for linear programming [33], the problem can also be solved in polynomial time.

V. Equivalent Network View Transformations

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

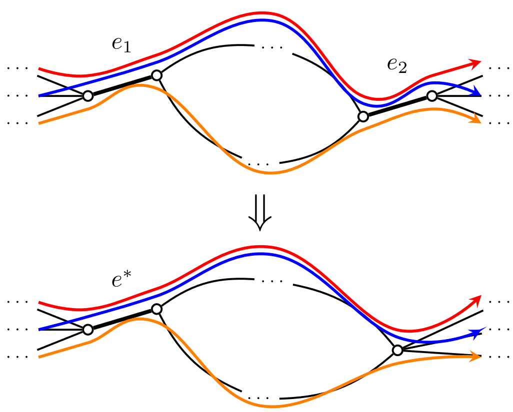


Fig. 5. Equivalent element aggregation.

Algorithm 1: Equivalent Element Aggregation

Input: $V(A, P, Q)$
Output: $V'(A', P', Q')$

1 **Function** EQUIVAGGREGATION*(V)*
2 $\quad \mathcal{V} \leftarrow \{V_j \mid V_j \leftarrow (A^j, P_j, Q_j), 1 \le j \le M\}$;
3 $\quad \mathcal{G} \leftarrow \text{GROUPBY}(\mathcal{V}, V_j \Rightarrow (v_j \leftarrow A^j, V_i))$;
4 $\quad$ **for** $G_j \in \mathcal{G}$ **do**
5 $\quad\quad V'_j \leftarrow \text{AGGREGATE}(v_j, \{V_{m_{j,1}}, \dots, V_{m_{j,|G_j|}}\})$;
6 $\quad M' \leftarrow |\mathcal{G}|$;
7 $\quad V' \leftarrow \left(\left[A'^1 \cdots A'^{M'}\right], \begin{bmatrix} P'_1 \\ \vdots \\ P'_{M'} \end{bmatrix}, \begin{bmatrix} Q'_1 \\ \vdots \\ Q'_{M'} \end{bmatrix} \right)$;
8 $\quad$ **return** V'
9 **Function** AGGREGATE*$(v_j, \{V_{m_{j,1}}, \dots, V_{m_{j,|G_j|}}\})$*
10 $\quad A'^j = v_j$;
11 $\quad P'_j \leftarrow \left[\bigoplus_i p_{m_{j,i},1}, \dots, \bigoplus_i p_{m_{j,i},K_i} \right]$;
12 $\quad Q'_j \leftarrow \left[\min_i q_{m_{j,i},1}, \dots, \min_i q_{m_{j,i},K_c} \right]$;
13 $\quad$ **return** (A'^j, P'_j, Q'_j)

A. Equivalent Element Aggregation

In this section, we introduce the *equivalent aggregation*. The example in Fig. 5 demonstrates the intuition: There are three flows with different colors and only they traverse both e_1 and e_2, so that we "aggregate" them together as a single element e^*. We give an algorithm in Algorithm 1 which guarantees that the resulted network view is equivalent to the original one. We analyze its efficiency and prove its correctness.

The network view is represented as row vectors (components), as demonstrated in Line 2. Line 3 groups the j-th *component* $V_j(A^j, P_j, Q_j)$ using the j-th row vector in A, A^j, as the key. M' denotes the number of groups, and the index of the k-th member in the j-th group is denoted as $m_{j,k}$. Line 5 computes the aggregation of the components in each group. Finally Line 7 constructs the new network view by merging all the aggregated components. For each component V_j, the time complexity for the grouping and the aggregation is $O(N(K_i + K_c))$ and $O(N(K_i + K_c))$ respectively while the MERGE process is totally logical, which yields a total time of $O(MN(K_i + K_c))$.

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

Theorem 3: $V' \leftarrow \text{EQUIVAGGREGATION}(V)$, $V' \sim V$.

Proof: Assume $g_{j,i}$ represents the index of the i-th components in G_j, and let $b_j \leftarrow \min_i m_{j,i}$ and $c_{j,k} \leftarrow \arg\min q_{m_{j,i},k}$.

First we check Equation 1a is met. Since $X = A \times P$, we have

$$
\begin{aligned}
& x_{i,k} \\
&= \bigoplus_j a_{i,j} \otimes p_{j,k} = \bigoplus_{1 \le j' \le M'} \left(\bigoplus_{1 \le l \le |G_{j'}|} a_{i,m_{j',l}} \otimes p_{m_{j',l},k} \right) \\
&= \bigoplus_{1 \le j' \le M'} \left(\bigoplus_{1 \le l \le |G_{j'}|} a_{i,b_{j'}} \otimes p_{m_{j',l},k} \right) \\
&= \bigoplus_{1 \le j' \le M'} a_{i,b_{j'}} \otimes \left(\bigoplus_{1 \le l \le |G_{j'}|} p_{m_{j',l},k} \right) \\
&= \bigoplus_{1 \le j' \le M'} a_{i,b_{j'}} \otimes p'_{j',k} = x'_{i,k}.
\end{aligned}
$$

The key steps are based on the facts that $\oplus$ is transitive and commutative, $\otimes$ is distributive over $\oplus$, and $\forall l \in [1, |G_{j'}|], a_{i,b_{j'}} = a_{i,m_{j',l}}$.

Now we check Equation 1b. We have

$$
\begin{aligned}
R &= \left\{ Y \mid A_j^{\mathsf{T}} Y^k \le q_{j,k}, \forall j, k \right\} \\
R' &= \left\{ Y \mid A'^{\mathsf{T}}_j Y^k \le q'_{j,k}, \forall j, k \right\} \\
&= \left\{ Y \mid A'^{\mathsf{T}}_{c_{j',k}} Y^k \le q_{c_{j',k},k}, \forall j', k \right\}.
\end{aligned}
$$

Since the constraints of R' is a subset of R, $R \subseteq R'$. If $R' \neq R$, $\exists \hat{Y} \in R' \setminus R$, meaning $\hat{Y}$ at least violates one constraint in R, say for $\hat{j}$ and $\hat{k}$, *i.e.*,

$$
A_{\hat{j}}^{\mathsf{T}} \hat{Y}^{\hat{k}} > q_{\hat{j},\hat{k}} \ge \min_l q_{m_{\hat{j}',l},\hat{k}} = q_{c_{\hat{j},\hat{k}},\hat{k}},
$$

which means $\hat{Y}$ also violates one constraint in R' and leads to contradiction with our assumption. So we have $R = R'$.

By Theorem 1, $V' \sim V$. $\square$

B. Equivalent Element Decomposition

In this section, we introduce the details of *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 ONV, however, the condition to aggregate components is not easy to be satisfied without further processing. Thus, we need to conduct another equivalent transformation in practice, namely *equivalent element decomposition*. An example of equivalent element decomposition is given in Fig. 6, where only the red flow traverses e_a, only the blue flow traverses e_b and only the red

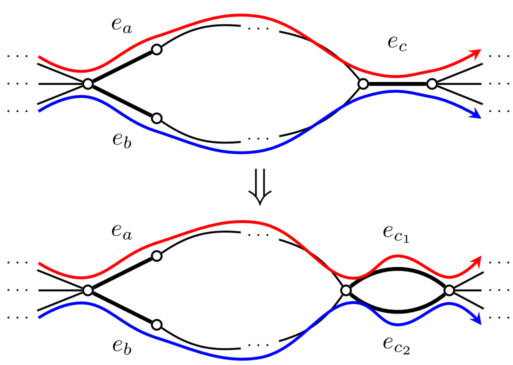


Fig. 6. Equivalent element decomposition.

and the blue flows traverse e_c. The three elements have the following metrics:

$$e_a : \textit{routingcost} = 1, \quad \textit{bandwidth} = 100\text{Mbps}$$
$$e_b : \textit{routingcost} = 2, \quad \textit{bandwidth} = 100\text{Mbps}$$
$$e_c : \textit{routingcost} = 3, \quad \textit{bandwidth} = 200\text{Mbps}$$
$$A^a = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad A^b = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad A^c = \begin{bmatrix} 1 \\ 1 \end{bmatrix}.$$

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

$$e_{c_1} : \textit{routingcost} = 3, \quad \textit{bandwidth} = 200\text{Mbps}$$
$$e_{c_2} : \textit{routingcost} = 3, \quad \textit{bandwidth} = 200\text{Mbps}$$
$$A^{c_1} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad A^{c_2} = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

After e_c is decomposed, we can invoke EQUIVAGGREGATION (Algorithm 1) and e_{c_1} and e_{c_2} can be aggregated with a and b respectively.

Theorem 4 gives the condition when an element can be safely decomposed without affecting the equivalence condition. The efficiency and privacy of equivalent decomposition depend on 1) how to identify redundant components, and 2) how to find the "basis". The first step corresponds to finding the minimal equivalent network view with only flow-correlated information, as discussed in Section IV-B.2, while the second step is similar to finding the minimal equivalent network view with flow-independent information, with the constraints that non-redundant network elements must be contained. Since the second step has been proved to be NP-Hard, in this paper, we use a heuristic approach which aims to simplify the selection of basis, as introduced in Algorithm 2.

Theorem 4: For $V_j(A^j, P_j, Q_j)$, we say V_j is redundant if and only if $A_j^{\mathsf{T}} Y^k \leq q_{j,k}$ is redundant for all k. If and only if V_j is redundant, we can construct an equivalent network view $V' = V \setminus V_j \cup \{V_{j,l}\}$ where V_j is decomposed as $V_{j,l}(A'^l, P'_l, Q'_l)$ with $A^j = \sum_l A'^l$, $P_j = P'_l$ and $Q_j = Q'_l$.

Proof: We still consider the criteria Equation 1a and Equation 1b and use the same symbols in Theorem 3.

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

$$\begin{aligned} x_{i,k} &= \bigoplus_u a_{i,u} \otimes p_{u,k} = \bigoplus_{u \neq j} a_{i,u} \otimes p_{u,k} + a_{i,j} \otimes p_{j,k} \\ &= \bigoplus_{u \neq j} a_{i,u} \otimes p_{u,k} + \left(\sum_l a'_{i,l} \right) \otimes p_{j,k} \\ &= \bigoplus_{u \neq j} a_{i,u} \otimes p_{u,k} + \bigoplus_l a'_{i,l} \otimes p_{j,k} \\ &= \bigoplus_{u \neq j} a_{i,u} \otimes p_{u,k} + \bigoplus_l a'_{i,l} \otimes p'_{l,k} = x'_{i,k}. \end{aligned}$$

For Equation 1b, first we consider the case when V_j is redundant but $V \nsim V'$. V_j is redundant so that $\forall k$, $A_j^{\mathsf{T}} X^k \leq q_{j,k}$ is redundant. According to Definition 3, we have feasible regions $R = R_j = R'$ for all k. Since we have already proved that Equation 1a holds, according to Theorem 1, $V \sim V'$ which leads contradiction.

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

Thus, we have proved that V_j can be equivalently decomposed if and only if V_j is redundant. $\square$

We introduce the concept of *dominance* of components. From Theorem 4, we can easily conclude that if an element can be decomposed, it dominates all the elements in the basis.

Definition 4 (Dominance of Components): We say a component $\boldsymbol{V_j}$ is *dominated* by another component $\boldsymbol{V_{j'}}$, if and only if, $\forall i$, $a_{i,j} \leq a_{i,j'}$.

Now we present the details of Algorithm 2. Line 2 identifies the set of decomposable components $\mathcal{D}$ according to Theorem 4, i.e. $\forall V_j \in \mathcal{D}$, V_j is redundant. In each iteration (Line 4-11), we try to decompose a decomposable elements into other network elements greedily, in the sense that $\forall l$, V_l is dominated by V_j, we decompose V_j to V_l and its complement. If the routing matrix for V_j is empty, it means V_j is decomposed to a set of V_ls. Otherwise, V_j cannot be

Algorithm 2: Equivalent Element Decomposition

Input: $V(A, P, Q)$
Output: $V'(A', P', Q')$

```
1  Function EquivDecomposition(V, F)
2      D ← FindEquivDecomposable(V)
3      V' ← V
4      foreach V_j ∈ D do
5          V' ← V' \ {V_j}
6          foreach V_l ∈ V' do
7              if V_j can be decomposed to V_l then
8                  V_l ← (A^l, P_l ⊕ P_j, Q_l)
9                  V_j ← (A^j − A^l, P_j, Q_j)
10         if A^j ≠ 0⃗ then
11             V' ← V' ∪ {V_j}
12     return V'
```

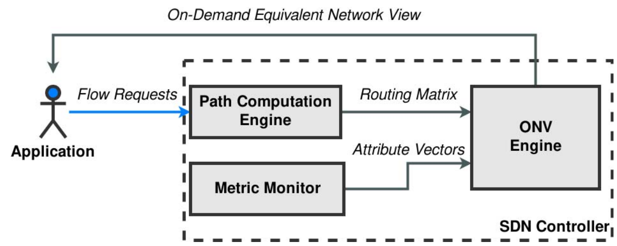


Fig. 7. The architecture of ONV.

decomposed to existing components so its remnant must be added back (Line 11).

Finally we invoke EQUIVAGGREGATION(V') to aggregate V'_j with the same A^j, which is also proved to maintain the equivalence condition as in Theorem 3. Thus, Algorithm 2 returns the equivalent network view.

For each iteration, the decomposition needs to check whether a component is dominated by another one, which may take $O(N)$ time and yield a total running time of $O(M^2N^2)$. Since we have assumed that flows only traverse simple paths, we encode the routing matrix for a component, which is a binary vector of size $N \times 1$, as a binary integer. This pre-processing step takes $O(MN)$ time. Thus, the dominance can be verified in $O(\log(N))$ time using bit `and` operation. The update on Line 8 takes only $O(K_i)$ time. The subtraction on Line 9 can be done using the bit `xor` operation, which also takes $O(\log(N))$ time. The routing matrix A^j can be reconstructed in $O(N)$ time in the outer loop. There are at most M^2 inner iterations and M outer iterations, so the total execution time is $O(M^2(\log(N) + K_i) + MN)$.

C. Privacy Enhancement

The equivalent aggregation and equivalent decomposition are equal to matrix transformations. While the application 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 and reduce information leak.

It is worth noting that ONV does not require applications to specify private information, e.g. private constraints and objective functions. Thus, it also protects the privacy of the applications.

D. System Implementation

The architecture of ONV is demonstrated in Fig. 7. User requests are first sent to the path computation engine, which obtains the routing matrix A. The ONV engine also pulls the attribute vectors, *i.e.*, P and Q from a monitoring component.

EQUIVAGGREGATION is first executed to avoid corner cases in finding redundant resource constraints. If no flow-correlated information is requested, the FINDEQUIVDECOMPOSABLE function returns all network elements. Otherwise, it uses the algorithm in [31] to find network elements with redundant constraints. Finally, EQUIVDECOMPOSITION is executed and the resulted equivalent network view is encoded as in Section III-E and returned to the user.

VI. EVALUATION

In this section, we evaluate ONV extensively to answer the following questions: 1) Can ONV achieve feasibility and optimality for heterogeneous objective functions? 2) How much can ONV simplify the network view? 3) How fast can ONV compute the abstract on-demand network view?

A. Experimental Setup

In this section, we introduce the general experimental setup of our evaluations and leave the methodologies for each experiment to the corresponding section.

Topology and Metrics: We use real-world topologies from two data sets: the topology zoo [34] and rocketfuel [35]. If a topology already has bandwidth information, we use the values 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. The values are multiplied by a given constant to avoid precision issues. For each topology, we generate three different routing cost distributions.

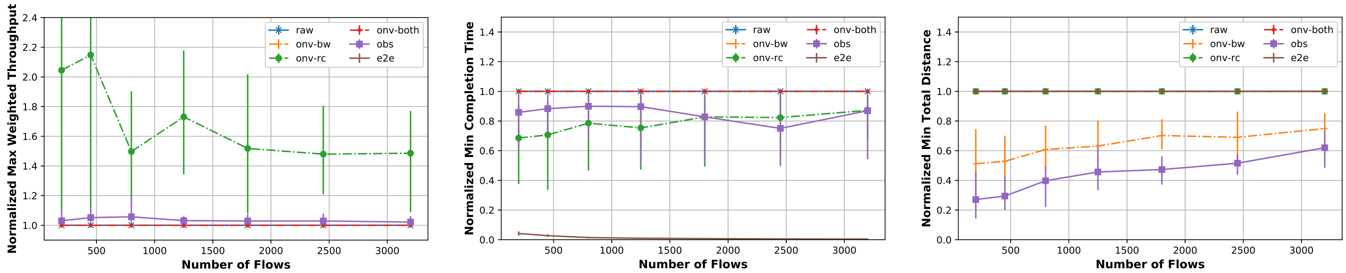
Algorithms to Find Redundant Constraints: To find the redundant network elements, we use the linear programming method introduced in [32].

Abstractions: We use six different network abstractions.

1) The *raw* network view is computed by the naive approach which contains all network elements on the paths.
2) Three various network views are computed by ONV with different guarantees. The *onv-bw* view only guarantees the equivalence of flow-correlated network resources. All decomposable network elements can be directly removed and its $\|V\|$ is equal to the number of non-decomposable network elements. The *onv-rc* view only guarantees the equivalence of flow-independent metrics so it considers all network elements to be decomposable, *i.e.*, Line 2 of Algorithm 2 returns V. The *onv-both* view guarantees equivalent network views where Line 2 of Algorithm 2 finds redundant network elements using an paralleled implementation of the algorithm in [31].
3) The *one-big-switch* (as in SDX [36]) view removes all network elements except the ingress/egress ones.
4) The *e2e* view (as in ALTO [21]) creates a virtual network element for each flow whose attributes are calculated with the variant routing algebra.

Flow Requests: We have 7 groups with different numbers of flows. There are three types of requests, depending on the metrics: *routingcost-only* (*rc*) *bandwidth-only* (*bw*), and *hybrid*. As the names suggest, they represent the cases where 1) only flow-independent metrics are requested, 2) only flow-correlated metrics are requested, and 3) both metric types are requested. The flows used in our benchmark are randomly generated based on the *server-client* communication pattern. We select a given subset of endpoints as servers, and for each server, we pick random endpoints as clients.

Runtime and Data Collection: The prototype system is built with Python and uses the PuLP framework. The *COIN Branch*



(a) (b) (c)

Fig. 8. Normalized results for different objective functions using different abstractions. (a) Results for weighted throughput (wpt). (b) Results for flow completion time (fct). (c) Results for total routing cost (trc).

and Cut (CBC) solver is used solve linear programming problems. The evaluations are emulated on a server with Linux kernel 3.19.0-25, 6 quad-core Intel(R) Xeon(R) E5-2620 v3 @2.40GHz CPU and 128 GB memory.

We collect the results for each ⟨topology+metrics, flow size+distribution, metric types⟩ combination. For each combination, we generate 10 different samples and calculate the average, standard deviation, the minimum and the maximum.

B. Optimality and Feasibility for Heterogeneous Objectives

To understand how different network views may have an impact on the optimality and feasibility of an application's objective functions, we conduct the following evaluation with multiple objective functions.

Objective Functions: We consider three objective functions from existing researches: 1) For the *wtp* objective function case, we give each flow a random weight and optimize the weighted throughput [37]. 2) For the *fct* objective function case, we give each flow a random data size and minimizes the total flow completion time [38]. 3) For the *trc* objective function case, we divide hosts into client and server groups. For each host in the client group, it selects the server node with the smallest routing cost. We sum the total routing cost as the value of the objective function.

Topology and Metrics: We use the Kdl topology (752 nodes, 1790 links) from the topology zoo. Since the coefficients of the objective functions are generated randomly, the values of objective functions may not be useful. Thus, we normalize the results as follows:

$$\text{Normalized} = \frac{\text{Optimal objective using a given view}}{\text{Optimal objective using } raw}.$$

The normalized results serve as indicators of whether the corresponding network view guarantees optimality and feasibility. Consider the optimization problem is to maximize a given objective function (as in *wpt*), if the normalized objective value $opt > 1$, it means that the objective value is larger than the real optimal value and thus the value is *infeasible*. On the other hand, if the normalized objective value $opt < 1$, it means that the objective value is smaller than the real optimal one and thus the value is *suboptimal*. For optimization problems that minimize a given objective function (*fct* and *trc*), the conclusion is the opposite. The conditions of whether a network view is *feasible* and *optimal* are listed in Table III.

TABLE III

CONDITIONS FOR FEASIBILITY AND OPTIMALITY

Objective function (normalized)	*Feasible*	*Optimal*
max weighted throughput (*wpt*)	$opt \leq 1$	$opt \geq 1$
min coflow completion time (*fct*)	$opt \geq 1$	$opt \leq 1$
min total routing cost (*trc*)	$opt \geq 1$	$opt \leq 1$

As demonstrated in Fig. 8, we can see that 1) one-big-switch abstraction can lead to *infeasible* solutions in all three cases; 2) end-to-end abstraction achieves both optimality and feasibility for *trc* but can lead to *infeasible* solutions for *wpt* and *fct* objective functions, indicating that it can provide accurate flow-independent information but very inaccurate information on flow-correlated resource (shared bottlenecks); 3) *onv-both* achieves both optimality and feasibility for all cases while the two variants also achieves optimality and feasibility for their targeted use cases, which indicates that ONV can provide accurate information on both flow-correlated information (*onv-bw* and *onv-both*), flow-independent information (*onv-rc* and *onv-both*) and the two types of metrics combined (*onv-both*).

C. Network Simplification

In this section, we demonstrate how much ONV can simplify the network and reduce the communication overhead with the following settings:

Metrics: We use the normalized $\|V\|$ to evaluate how much a network view is simplified and use the number of bytes in the encoded JSON string to evaluate the communication overhead of an abstract network view.

As we can see form Fig. 9, ONV can simplify the network significantly. Specifically, the *bandwidth-equivalent* network view only uses less than 40% of the network elements in the original one for all six topologies except *Colt*. It even uses less network elements than the *one-big-switch* abstraction in certain topologies. While the *routingcost-equivalent* and the *completely equivalent* network views typically contain more elements, they can still reduce 50% to 80% of the network information for 200 flows, and 20% to 60% of the network information even for more than 3,000 flows.

We also analyze the factors that may determine the simplification results. From Fig. 9, we can see that the number of elements increases as the number of flows increases. Thus, we consider the largest flow requests, *i.e.*, with 3,200 flows

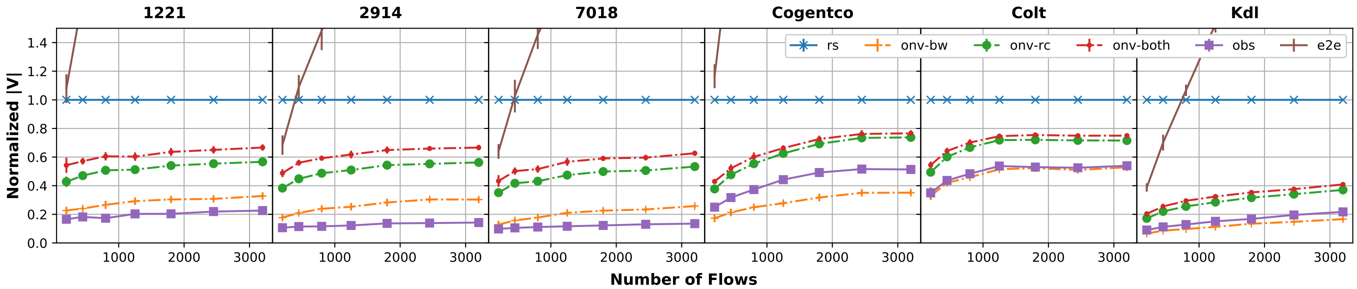
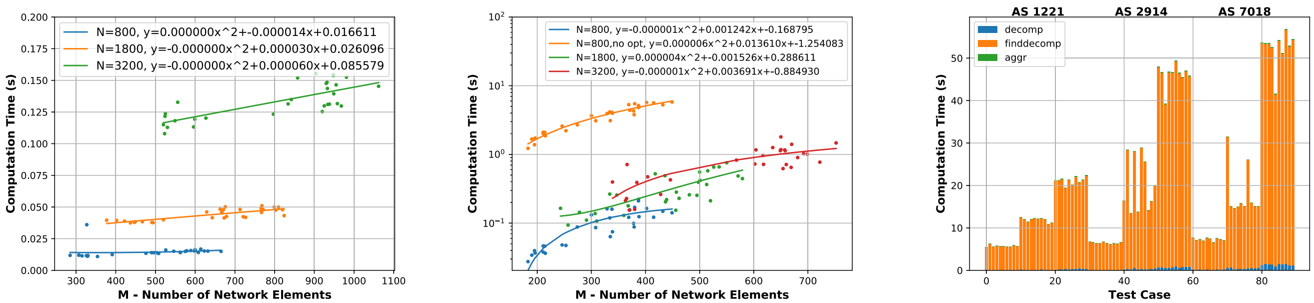


Fig. 9. Normalized $\|V\|$ for different topologies.

TABLE IV

TOPOLOGIES AND THE ABSOLUTE RESULTS FOR 3200 FLOW REQUESTS

Topology	*#Nodes*	*#Edges*	Original View (*raw*)		*Bandwidth-equivalent* View (*onv-bw*)		*End-to-end-equivalent* View (*onv-rc*)		*Equivalent* View (*onv-both*)	
			$\|V\|$	Relative size	$\|V\|$	Relative size	$\|V\|$	Relative size	$\|V\|$	Relative size
AS 1221	5023	6259	566.00±40.63	-	185.40±14.39	0.32	320.80±23.69	0.57	377.20±26.21	0.67
AS 2914	10820	16422	883.90±59.85	-	267.80±20.74	0.30	497.50±40.67	0.56	589.10±42.70	0.67
AS 7018	20593	25199	919.60±25.61	-	236.90±13.32	0.27	490.70±17.22	0.56	576.40±21.86	0.65
Cogentco	197	243	285.90±17.36	-	100.30±5.70	0.35	210.60±8.87	0.74	218.90±10.98	0.77
Colt	153	177	172.30±5.81	-	91.10±3.60	0.53	123.30±4.60	0.71	129.20±5.03	0.75
Kdl	754	895	955.70±39.25	-	159.00±9.68	0.17	354.30±15.34	0.37	390.60±17.40	0.41



(a) (b) (c)

Fig. 10. Computation time. (a) Computation time of EQUIVAGGREGATION. (b) Computation time of EQUIVDECOMPOSITION. (c) Computation time breakdown.

and summarize the absolute values in Table IV.[3] The relative size of a given view is calculated as proportion of network elements compared with the one in the *raw* view, *i.e.*,

$$\text{relative size} = \frac{\|V\| \text{ of a given view}}{\|V\| \text{ of the } \textit{raw} \text{ view}}.$$

As shown in Table IV, the network elements revealed in abstract network views are much less than the ones contained in the topologies, suggesting that on-demand network views can protect the privacy of network providers while providing useful information to network-aware adaptive applications.

D. Computation Time

Methodology: The time complexity depends on both the number of flows N and the number of network elements M. We choose 3 different numbers of flows: 800 (20 servers and 40 clients per server), 1800 (30 servers and 60 clients per server) and 3200 (40 servers and 80 clients per server). For each M, we generate 10 samples for topologies AS 1221, AS 2914 and AS 7018. For equivalent decomposition, we also turn on/off the binary code optimization.

[3]Edges are bidirectional so the total number of elements in a topology is twice the number of edges.

As we can see in Fig. 10(a), the time curve fits well with a linear function of the number of network elements M. The coefficients of x also roughly grows linearly as the number of flows N, which demonstrates the time complexity analysis for Algorithm 1 is correct.

Also, we can also see that in Fig. 10(b), the time curve also fits with a quadratic function of M. While we cannot derive directly from the coefficients that the time complexity is linear with the logarithm of N, the comparison to an unoptimized implementation (*i.e.*, using linear scan as in Definition 4) demonstrates an improvement of 40 to 60.

Finally we demonstrate how much each component contributes to the total execution time. As we can see in Fig. 10(c), the total execution is less than a minute even for resource reservation for a lot of flows in very large scale networks. In particular, both EQUIVAGGREGATION (denoted as *aggr*) and EQUIVDECOMPOSITION (denoted as *decomp*) only take a very small proportion. While the time is sufficient to traditional traffic engineering which may take hours or days, the operator can optionally skip the equivalent decomposition procedure if the application-layer scheduler demands smaller traffic engineering intervals.

E. Summary

In this section, we evaluate the performance of ONV thoroughly. We demonstrate that ONV guarantees both feasibility and optimality for heterogeneous objective functions. ONV can substantially reduce the number of leaked information and also the communication overhead (up to 4.5x improvement). It can produce an abstract on-demand network view within a minute for very large scale networks (>20,000 nodes and >25,000 edges) and flow requests (>3,000 flows). Thus, ONV effectively enables collaborative optimization with non-administrative network-aware adaptive applications.

VII. 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 [39]–[41]. Previous studies [42] 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 [38], [43], [44] 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 [1] system and global data intensive science networks [26]. Feeding these applications with more accurate network information allows them to make more intelligent operating decisions. Network information is also used to optimize video streaming for multiple objectives, such as QoE [45] and fairness [46].

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) [36] 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 [27], [29].

SOL [47] and CoFlow [38] 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 [48]–[50] typically protects the privacy by multiplying a monomial matrix. ONV 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 [17]–[20] 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 [51]. Modern SDN controllers [9]–[12] also provide the global view using the annotated graph model.

The hose model [15] 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 [52], [53]. SDX also uses this model to encapsulate the underlying network topology. Data center fabrics are highly customized for scalability [16] and can be modeled as a non-blocking switch where congestion only occurs on access links [23], thus, the *one-big-switch* abstraction is also widely used for data center flow scheduling and tenant resource provisioning [38], [43], [44].

The mesh model is mostly used by web-based applications or measurement frameworks, which have no control over the network. The mesh model consists of several flows (host pairs) and provides a single mesh for each flow (pair) with the associated metrics. PerfSONAR [54], Meridian [55] and ClosestNode [56] are some concrete examples which provide such end-to-end network views based on measurement, while P4P [42] and the ALTO (Application-Layer Traffic Optimization) protocol [21] are leveraging the network providers' information.

ONV is similar to ALTO in the sense that in both cases information is provided by the network to non-administrative applications, 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 ONV from other (especially QoS related) routing protocols and network views based on topological aggregation [19].

Recently Nikolenko *et al.* [57] has introduced an algorithm to simplify the network topologies, which is also based on the principle of *equivalence* and equivalent network transformations. Compared with their work, we have a different definition of equivalence originated from the applications' perspective. While their work is still transforming the topology, our equivalent transformations are based on a more abstracted network representation which allows us to conduct more sophisticated transformations beyond the topological constraints. Similar ideas are also applied in some newer researches [58], [59].

VIII. Conclusion

In this paper, we systematically study the problem of providing an accurate on-demand network view for application-layer multi-flow optimization. Our abstraction is based on the principle of *equivalence* which guarantees generality, feasibility and optimality. We design the ONV framework to construct equivalent network views and evaluate its performance compared with some well-known network view abstractions. Currently, ONV leverages the SDN technology to provide the network view service in a centralized way, and leaves distributed on-demand network view as a future extension.

ACKNOWLEDGMENT

The authors would like to thank C. Gu, J. Zhang, S. Chen, X. Lin, H. Wang, and H. Du for their help during the preparation of the paper. This paper is an extension to a conference paper [60], which is also published as a poster earlier [61]. They would also like to thank the anonymous TON reviewers for their valuable feedback. 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.

REFERENCES

[1] S. Jain *et al.*, "B4: Experience with a globally-deployed software defined wan," in *Proc. ACM SIGCOMM Conf. (SIGCOMM)* New York, NY, USA, 2013, pp. 3–14.

[2] *Official Public Website for the ATLAS Experiment at CERN*, CERN, Meyrin, Switzerland, 2018. [Online]. Available: https://home.cern/science/experiments/cms

[3] *The Compact Muon Solenoid Experiment*, CERN, ATLAS Exp., Geneva, Switzerland, 2018. [Online]. Available: https://atlas.cern/

[4] *AT&T Vision Alignment Challenge Technology Survey—AT&T Domain 2.0 Vision White Paper*," AT&T, Dallas, TX, USA, 2013.

[5] *OSCARS*, Lawrence Berkeley Nat. Lab., Energy Sci. Netw., Berkeley, CA, USA, 2018. [Online]. Available: https://www.es.net/engineering-services/oscars/

[6] R. Viswanathan, G. Ananthanarayanan, and A. Akella, "CLARINET: WAN-aware optimization for analytics queries," in *Proc. 12th USENIX Symp. Oper. Syst. Design Implement. (OSDI)*. Savannah, GA, USA: USENIX Association, 2016, pp. 435–450.

[7] M. Chowdhury, Y. Zhong, and I. Stoica, "Efficient coflow scheduling with Varys," in *Proc. ACM Conf. SIGCOMM*, New York, NY, USA, 2014, pp. 443–454.

[8] J. Rehn *et al.*, "PhEDEx high-throughput data transfer management system," in *Proc. Comput. High Energy Nucl. Phys. (CHEP)*, 2006, pp. 1–4.

[9] N. Gude *et al.*, "NOX: Towards an operating system for networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 38, no. 3, pp. 105–110, 2008.

[10] T. Koponen *et al.*, "Onix: A distributed control platform for large-scale production networks," in *Proc. OSDI*, vol. 10, 2010, pp. 1–6.

[11] P. Berde *et al.*, "ONOS: Towards an open, distributed SDN OS," in *Proc. 3rd Workshop Hot Topics Softw. Defined Netw. (HotSDN)*, New York, NY, USA, 2014, pp. 1–6.

[12] J. Medved, R. Varga, A. Tkacik, and K. Gray, "OpenDaylight: Towards a model-driven SDN controller architecture," in *Proc. IEEE Int. Symp. World Wireless, Mobile Multimedia Netw.*, Jun. 2014, pp. 1–6.

[13] S. Gao, Z. Peng, B. Xiao, A. Hu, and K. Ren, "FloodDefender: Protecting data and control plane resources under SDN-aimed DoS attacks," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.

[14] Y. Rekhter, T. Li, and S. Hares, "*A Border Gateway Protocol 4 (BGP-4)*, document RFC 4271, 2005.

[15] N. G. Duffield *et al.*, "A flexible model for resource management in virtual private networks," in *Proc. ACM Conf. Appl., Technol., Archit., Protocols Comput. Commun. (SIGCOMM)*, New York, NY, USA, 1999, pp. 95–108.

[16] A. Greenberg *et al.*, "VL2: A scalable and flexible data center network," in *Proc. SIGCOMM ACM SIGCOMM Conf. Data Commun.*, New York, NY, USA, 2009, pp. 51–62.

[17] J. Moy, *OSPF Version 2*, document RFC 2178, 1997. [Online]. Available: https://tools.ietf.org/html/rfc2178

[18] D. Oran, *OSI IS-IS Intra-domain Routing Protocol*, document RFC 1142, 1990.

[19] T. Vu, A. Baid, H. Nguyen, and D. Raychaudhuri, "EIR: Edge-aware interdomain routing protocol for the future mobile Internet," WINLAB, Rutgers Univ., New Brunswick, NJ, USA, Tech. Rep. WINLAB-TR-414, 2013.

[20] W. C. Lee, "Topology aggregation for hierarchical routing in ATM networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 25, no. 2, pp. 82–92, 1995.

[21] R. Alimi, Y. Yang, and R. Penno, *Application-Layer Traffic Optimization (ALTO) Protocol*, document RFC 7285, 2014.

[22] S. Uludag, K.-S. Lui, K. Nahrstedt, and G. Brewster, "Analysis of topology aggregation techniques for QoS routing," *ACM Comput. Surv.*, vol. 39, no. 3, p. 7, 2007.

[23] H. Rodrigues, J. R. Santos, Y. Turner, P. Soares, and D. O. Guedes, "Gatekeeper: Supporting bandwidth guarantees for multi-tenant datacenter networks," in *Proc. WIOV*, 2011, pp. 1–8.

[24] A. Kumar *et al.*, "BwE: Flexible, hierarchical bandwidth allocation for WAN distributed computing," in *Proc. ACM Conf. Special Interest Group Data Commun. (SIGCOMM)*, New York, NY, USA, 2015, pp. 1–14.

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

[26] E. Dart, L. Rotman, B. Tierney, M. Hester, and J. Zurawski, "The science DMZ: A network design pattern for data-intensive science," *Sci. Program.*, vol. 22, no. 2, pp. 173–185, 2014.

[27] 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 *Proc. IEEE 23rd Int. Symp. Qual. Service (IWQoS)*, Jun. 2015, pp. 261–266.

[28] M. Rabbat, R. Nowak, and M. Coates, "Multiple source, multiple destination network tomography," in *Proc. IEEE INFOCOM*, vol. 3, Mar. 2004, pp. 1628–1639.

[29] J. L. Sobrinho, "Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet," *IEEE/ACM Trans. Netw.*, vol. 10, no. 4, pp. 541–550, Aug. 2002.

[30] S. A. Vavasis, "On the complexity of nonnegative matrix factorization," *SIAM J. Optim.*, vol. 20, no. 3, pp. 1364–1377, Jan. 2010.

[31] J. Telgen, "Identifying redundant constraints and implicit equalities in systems of linear constraints," *Manage. Sci.*, vol. 29, no. 10, pp. 1209–1222, 2002.

[32] S. Paulraj and P. Sumathi, "A comparative study of redundant constraints identification methods in linear programming problems," *Math. Problems Eng.*, vol. 2010, Sep. 2010, Art. no. 723402.

[33] N. Karmarkar, "A new polynomial-time algorithm for linear programming," in *Proc. 16th Annu. ACM Symp. Theory Comput.*, 1984, pp. 302–311.

[34] S. Knight, H. X. Nguyen, N. Falkner, R. Bowden, and M. Roughan, "The Internet topology zoo," *IEEE J. Sel. Areas Commun.*, vol. 29, no. 9, pp. 1765–1775, Oct. 2011.

[35] N. Spring, R. Mahajan, D. Wetherall, and T. Anderson, "Measuring ISP topologies with Rocketfuel," *IEEE/ACM Trans. Netw.*, vol. 12, no. 1, pp. 2–16, Feb. 2004.

[36] A. Gupta *et al.*, "SDX: A software defined Internet exchange," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 4, pp. 551–562, 2015.

[37] Y. Bartal *et al.*, "Online competitive algorithms for maximizing weighted throughput of unit jobs," in *Proc. Annu. Symp. Theor. Aspects Comput. Sci.* Berlin, Germany: Springer, 2004, pp. 187–198.

[38] M. Chowdhury and I. Stoica, "Coflow: A networking abstraction for cluster applications," in *Proc. 11th ACM Workshop Hot Topics Netw. (HotNets-XI)*, New York, NY, USA, 2012, pp. 31–36.

[39] R. K. Mok, E. W. W. Chan, and R. K. C. Chang, "Measuring the quality of experience of HTTP video streaming," in *Proc. 12th IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM) Workshops*, May 2011, pp. 485–492

[40] N. Zhang, Y. Lee, M. Radhakrishnan, and R. K. Balan, "GameOn: P2P gaming on public transport," in *Proc. MobiSys*, 2015, pp. 105–119.

[41] Y. Lee, S. Agarwal, C. Butcher, and J. Padhye, "Measurement and estimation of network QoS among peer Xbox 360 game players," in *Proc. Int. Conf. Passive Active Netw. Meas.* Berlin, Germany: Springer, 2008, pp. 41–50.

[42] H. Xie, Y. R. Yang, A. Krishnamurthy, Y. G. Liu, and A. Silberschatz, "P4P: Provider portal for applications," in *Proc. ACM SIGCOMM Conf. Data Commun. (SIGCOMM)* New York, NY, USA, 2008, pp. 351–362.

[43] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, and A. Vahdat, "Hedera: Dynamic flow scheduling for data center networks," in *Proc. NSDI*, vol. 10, 2010, p. 19.

[44] M. Alizadeh *et al.*, "pFabric: Minimal near-optimal datacenter transport," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, New York, NY, USA, 2013, pp. 435–446.

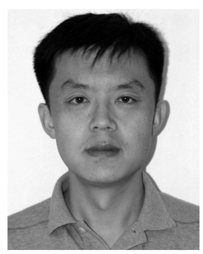
[45] P. Georgopoulos, Y. Elkhatib, M. Broadbent, M. Mu, and N. Race, "Towards network-wide QoE fairness using openflow-assisted adaptive video streaming," in *Proc. ACM SIGCOMM Workshop Future Hum.-Centric Multimedia Netw. (FhMN)*, New York, NY, USA, 2013, pp. 15–20.
[46] G. Cofano *et al.*, "Design and performance evaluation of network-assisted control strategies for HTTP adaptive streaming," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 13, no. 3s, pp. 42:1–42:24, Jun. 2017.
[47] V. Heorhiadi, M. K. Reiter, and V. Sekar, "Simplifying software-defined network optimization using SOL," in *Proc. 13th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*, 2016, pp. 223–237.
[48] Y. Hong, "Privacy-preserving collaborative optimization," Ph.D. dissertation, Dept. Manage. Sci. Inf. Syst., Rutgers Univ., New Brunswick, NJ, USA, 2013.
[49] J. Vaidya, "Privacy-preserving linear programming," in *Proc. ACM Symp. Appl. Comput. (SAC)*, New York, NY, USA, 2009, pp. 2002–2007.
[50] J. Li and M. J. Atallah, "Secure and private collaborative linear programming," in *Proc. Collaborative Comput., Netw., Appl. Worksharing, IEEE CollaborateCom Int. Conf.*, Nov. 2006, pp. 1–8.
[51] J. Medved *et al.*, *A Data Model for Network Topologies*, document draft-ietf-i2rs-yang-network-topo-20.txt, Internet Engineering Task Force, Internet-Draft draft-ietf-i2rs-yang-network-topo-11, Feb. 2017.
[52] C. Monsanto, J. Reich, N. Foster, J. Rexford, and D. Walker, "Composing software defined networks," in *Proc. 10th USENIX Symp. Netw. Syst. Design Implement. (NSDI)*. Lombard, IL, USA: USENIX Association, 2013, pp. 1–13.
[53] A. Voellmy, J. Wang, Y. R. Yang, B. Ford, and P. Hudak, "Maple: Simplifying SDN programming using algorithmic policies," in *Proc. ACM SIGCOMM Conf. SIGCOMM*, New York, NY, USA, 2013, pp. 87–98.
[54] A. Hanemann *et al.*, "PerfSONAR: A service oriented architecture for multi-domain network monitoring," in *Proc. Int. Conf. Service-Oriented Comput.* Berlin, Germany: Springer, 2005, pp. 241–254.
[55] B. Wong, A. Slivkins, and E. G. Sirer, "Meridian: A lightweight network location service without virtual coordinates," in *Proc. Conf. Appl., Technol., Archit., Protocols Comput. Commun. (SIGCOMM)*, New York, NY, USA, 2005, pp. 85–96.
[56] B. Wong and E. G. Sirer, "ClosestNode.Com: An open access, scalable, shared geocast service for distributed systems," *SIGOPS Oper. Syst. Rev.*, vol. 40, no. 1, pp. 62–64, Jan. 2006.
[57] S. I. Nikolenko, K. Kogan, and A. F. Anta, "Network simplification preserving bandwidth and routing capabilities," in *Proc. IEEE INFOCOM Conf. Comput. Commun.*, May 2017, pp. 1–9.
[58] Q. Xiang *et al.*, "Optimizing in the dark: Learning an optimal solution through a simple interface," in *Proc. AAAI*, Nov. 2018.
[59] Q. Xiang *et al.*, "Fine-grained, multi-domain network resource abstraction as a fundamental primitive to enable high-performance, collaborative data sciences," in *Proc. Int. Conf. High Perform. Comput, Netw., Storage, Anal. (SC)*, Piscataway, NJ, USA: IEEE Press, 2018, pp. 5:1–5:13.
[60] K. Gao, Q. Xiang, X. Wang, Y. R. Yang, and J. Bi, "NOVA: Towards on-demand equivalent network view abstraction for network optimization," in *Proc. IEEE/ACM 25th Int. Symp Qual. Service (IWQoS)*, Jun. 2017, pp. 1–10.
[61] K. Gao *et al.*, "ORSAP: Abstracting routing state on demand," in *Proc. IEEE 24th Int Conf. Netw. Protocols (ICNP)*, Nov. 2016, pp. 1–2.

Kai Gao received the B.S. and Ph.D. degrees from the Department of Computer Science and Technology, Tsinghua University, Beijing, China, in 2018.

He is currently an Assistant Research Scientist with the College of Cybersecurity, Sichuan University. His research interests include programming languages and distributed runtime systems for newly emerged networking techniques such as software-define networking and network function virtualization.

Qiao Xiang received the bachelor's degree in information security and the bachelor's degree in economics from Nankai University in 2007, and the master's and Ph.D. degrees in computer science from Wayne State University in 2012 and 2014, respectively. From 2014 to 2015, he was a Post-Doctoral Fellow with the School of Computer Science, McGill University. He is currently an Associate Research Scientist with the Department of Computer Science, Yale University. His research interests include software defined networking, resource discovery and orchestration in collaborative data sciences, interdomain routing, and wireless cyber-physical systems.

Xin Wang received the B.S. degree in computer science from Tongji University, Shanghai, China, in 2013, where he is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology. He is also with the Key Laboratory of Embedded System and Service Computing, Ministry of Education, Beijing, China. His research interests include computer networks, software-defined networks, and distributed computing.



Yang Richard Yang received the B.E. degree in computer science and technology from Tsinghua University in 1993, and the M.S. and Ph.D. degrees in computer science from the University of Texas at Austin in 1998 and 2001, respectively. He is currently a Professor of computer science and electrical engineering with Yale University. His work has been implemented/adopted in products/systems of major companies (e.g., AT&T, Alcatel-Lucent, Cisco, Google, Microsoft, and Youku) and featured in mainstream media, including *Economist*, *Forbes*, *Guardian*, *Information Week*, *MIT Technology Review*, *Science Daily, USA Today, Washington Post,* and *Wired*, among others. His research was supported by both the U.S. government funding agencies and leading industrial corporations. His research interests span areas including computer networks, mobile computing, wireless networking, and network security. His awards include the CAREER Award from the National Science Foundation and the Google Faculty Research Award.

Jun Bi (S'98–A'99–M'00–SM'14) received the B.S., C.S., and Ph.D. degrees from the Department of Computer Science, Tsinghua University, Beijing, China. He is currently a Changjiang Scholar Distinguished Professor with Tsinghua University, where he serves as the Director of the Network Architecture Research Division and the Deputy Dean of the Institute for Network Sciences and Cyberspace. He is also the Director of the Future Network Theory and Application Research Division, Beijing National Research Center for Information Science and Technology. He has published over 200 research papers and 20 Internet RFCs or drafts. He held 30 innovation patents. He has successfully led tens of research projects. His current research interests include Internet architecture, SDN/NFV, and network security. He is a Distinguished Member of the China Computer Federation.