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

Kai Gao[23], Qiao Xiang[13], Xin Wang[13], Yang Richard Yang[13] and Jun Bi[2]

[1]Department of Computer Science, Tongji University
[2]Institute for Network Science and Cyberspace, Tsinghua University
[3]Department of Computer Science, Yale University

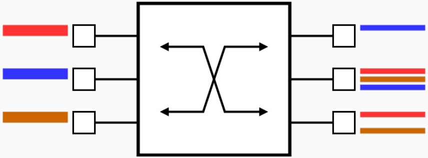
Table of Contents

1. Introduction
2. NOVA
3. Summary

Introduction

A Motivating Example

- Three requests: red (r), blue (b) and brown (y)

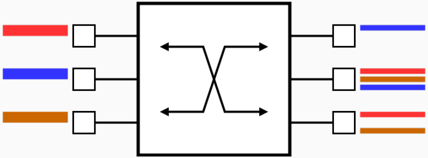


A Motivating Example

- Three requests: red (r), blue (b) and brown (y)
- Private QoS function:

$$\frac{1}{\alpha \times \mathit{hopcount} + \frac{\beta}{\mathit{bandwidth}}}$$

- Local preference: red request is from a privileged client



Existing Work

- Centralized optimization framework for SDN
    - Example: SOL[1]
    - Limitation: Require applications to submit private information

---

[1] Victor Heorhiadi, Michael K Reiter, and Vyas Sekar. "Simplifying Software-Defined Network Optimization Using SOL", NSDI'16

Existing Work

- Centralized optimization framework for SDN
- Global view
    - Example: NOX[1]
    - Limitation: Contain redundant information even after filtering, leak sensitive network information (topology),

---

[1] Natasha Gude et al. “NOX: towards an operating system for networks”, SIGCOMM CCR’08

Existing Work

- Centralized optimization framework for SDN
- Global view
- One-Big Switch[1]
    - Example: CoFlow[2], VL2[3]
    - Limitation: Assume no bottleneck inside the network, cannot present end-to-end information

---

[1] Nanxi Kang et al. “Optimizing the “one big switch” abstraction in software-defined networks”, CoNEXT’13

[2] Mosharaf Chowdhury and Ion Stoica. “Coflow: A networking abstraction for cluster applications”, HotNet’12

[3] Albert Greenberg et al. “VL2: a scalable and flexible data center network”, SIGCOMM’09

Existing Work

- Centralized optimization framework for SDN
- Global view
- One-Big Switch
- End-to-end map abstraction
  - Example: ALTO[1]
  - Limitation: Cannot present shared bottleneck

---

[1] Richard Alimi, Yang Yang, and Reinaldo Penno. “Application-layer traffic optimization (ALTO) protocol”, RFC 7285

Overview

Requirements:

- *Constructed on demand*
- *Without loss of information*
- *Protecting sensitive information*
- *Compact in size*

Overview

Requirements:

- Constructed on demand
- Without loss of information
- Protecting sensitive information
- Compact in size

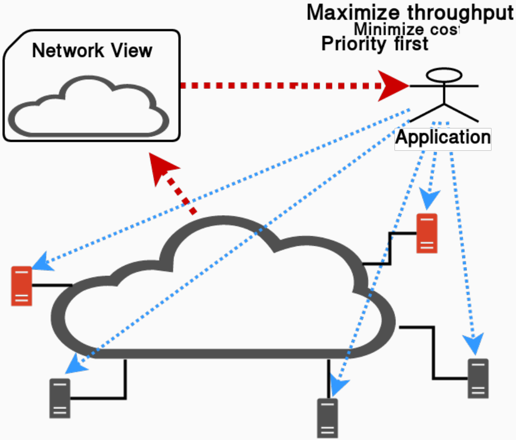
The NOVA abstraction is

- Based on flow queries
- Using equivalent transformation
- Using irreversible transformation
- Reducing redundant information as mush as possible

NOVA

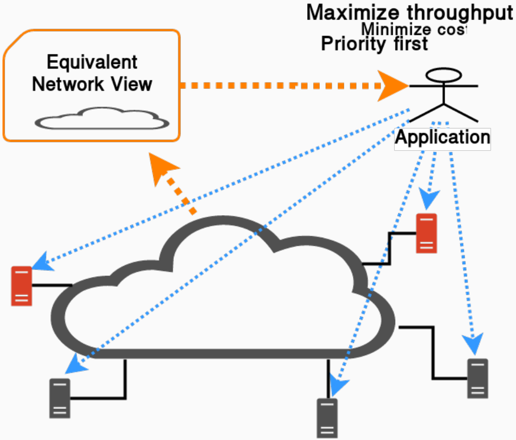
Key Observation

- Observation: Applications use network views to make traffic scheduling decisions to achieve a certain **objective**.

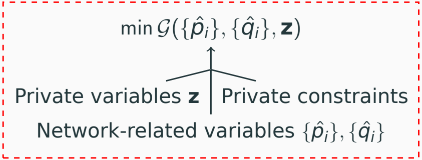


Key Observation

- Observation: Applications use network views to make traffic scheduling decisions to achieve a certain **objective**.
- Conclusion: Equivalent network views allow applications to always make the *same* optimal decision.



Application-Network Collaborative Optimization Model



$\min \mathcal{G}(\{\hat{p}_i\}, \{\hat{q}_i\}, \mathbf{z})$

Private variables $\mathbf{z}$

Private constraints

Network-related variables $\{\hat{p}_i\}, \{\hat{q}_i\}$

F: set of flows

Application

Network

Routing policies, topology, etc.

Application objective function template:

$$\min \mathcal{E}(\{\hat{p}_i\}, \{\hat{q}_i\})$$

Application-Network Collaborative Optimization Model

$\min \mathcal{E}(\{\hat{p}_i\}, \{\hat{q}_i\})$

F: set of flows

Application

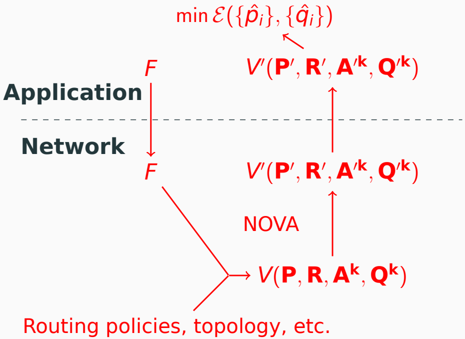
Network

Routing policies, topology, etc.

Application objective function template:

$$\min \mathcal{E}(\{\hat{p}_i\}, \{\hat{q}_i\})$$

Application-Network Collaborative Optimization Model



$\min \mathcal{E}(\{\hat{p}_i\}, \{\hat{q}_i\})$

F

$V'(\mathbf{P}', \mathbf{R}', \mathbf{A}'^{\mathbf{k}}, \mathbf{Q}'^{\mathbf{k}})$

Application

Network

F

$V'(\mathbf{P}', \mathbf{R}', \mathbf{A}'^{\mathbf{k}}, \mathbf{Q}'^{\mathbf{k}})$

NOVA

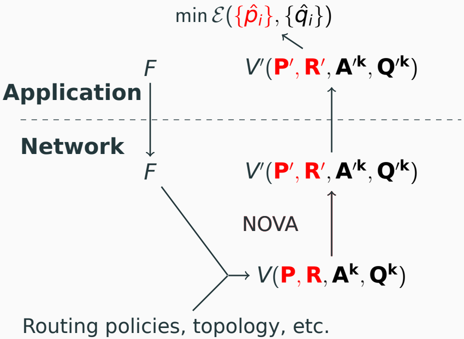
$V(\mathbf{P}, \mathbf{R}, \mathbf{A}^{\mathbf{k}}, \mathbf{Q}^{\mathbf{k}})$

Routing policies, topology, etc.

Network processing:

- Take flow request F and return a subset of global view V
- NOVA kicks in and produces the equivalent view V'

Application-Network Collaborative Optimization Model



$\min \mathcal{E}(\{\hat{p}_i\}, \{\hat{q}_i\})$

F

Application

Network

F

$V'(\mathbf{P}', \mathbf{R}', \mathbf{A}'^{\mathbf{k}}, \mathbf{Q}'^{\mathbf{k}})$

$V'(\mathbf{P}', \mathbf{R}', \mathbf{A}'^{\mathbf{k}}, \mathbf{Q}'^{\mathbf{k}})$

NOVA

$V(\mathbf{P}, \mathbf{R}, \mathbf{A}^{\mathbf{k}}, \mathbf{Q}^{\mathbf{k}})$

Routing policies, topology, etc.

End-to-End metrics ($\hat{p}_i$):

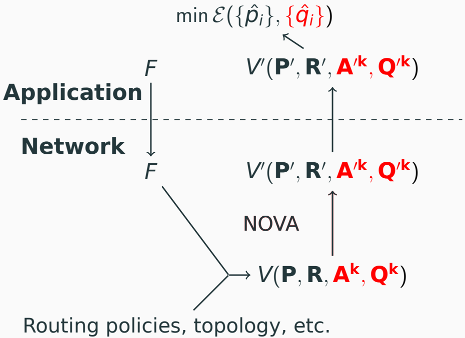
- Accumulated along the forwarding paths
- Values are (statistically) independent of flows
- Based on Variant QoS metric algebra: extended from *Sobrinho's QoS algebra*[2], relaxed path concatenation to linearize QoS metrics

$$\hat{\mathbf{P}} = \mathbf{R} \times \mathbf{P}$$

---

[2] João Luís Sobrinho. "Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet", TON'02

Application-Network Collaborative Optimization Model



$\min \mathcal{E}(\{\hat{p}_i\}, \{\hat{q}_i\})$

Application

F

$V'(\mathbf{P}', \mathbf{R}', \mathbf{A}'^{\mathbf{k}}, \mathbf{Q}'^{\mathbf{k}})$

Network

F

$V'(\mathbf{P}', \mathbf{R}', \mathbf{A}'^{\mathbf{k}}, \mathbf{Q}'^{\mathbf{k}})$

NOVA

$V(\mathbf{P}, \mathbf{R}, \mathbf{A}^{\mathbf{k}}, \mathbf{Q}^{\mathbf{k}})$

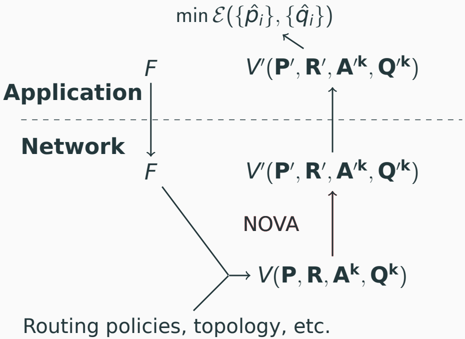
Routing policies, topology, etc.

Shared resources ($\hat{q}_i$):

- Constrained by resources available on the forwarding paths
- Shared by different flows

$$\mathbf{A}^k \times \hat{\mathbf{q}}^k \leq \mathbf{q}^k$$

Application-Network Collaborative Optimization Model



$\min \mathcal{E}(\{\hat{p}_i\}, \{\hat{q}_i\})$

F $V'(\mathbf{P}', \mathbf{R}', \mathbf{A}'^{\mathbf{k}}, \mathbf{Q}'^{\mathbf{k}})$

Application

Network

F $V'(\mathbf{P}', \mathbf{R}', \mathbf{A}'^{\mathbf{k}}, \mathbf{Q}'^{\mathbf{k}})$

NOVA

$V(\mathbf{P}, \mathbf{R}, \mathbf{A}^{\mathbf{k}}, \mathbf{Q}^{\mathbf{k}})$

Routing policies, topology, etc.

Common constraints:

$$\mathbf{R} \geq 0, \mathbf{A}^k \geq 0, \mathbf{q}^k \geq 0, \hat{\mathbf{q}}^k \geq 0$$

Equivalent Network View

A declarative definition (what we want to achieve):

- Applications can use the view to make the same traffic scheduling decision.

Equivalent Network View

A declarative definition (what we want to achieve):

- Applications can use the view to make the same traffic scheduling decision.

A mathematical definition (how we verify the equivalence):

- End-to-End metrics:

$$\mathbf{R_1} \times \mathbf{P_1} = \mathbf{R_2} \times \mathbf{P_2}$$

- Shared resources:

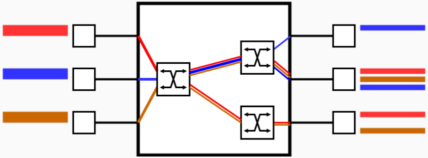
$$F_1^k = \left\{ \mathbf{x} \mid \mathbf{A_1^k} \mathbf{x} \leq \mathbf{q_1^k} \right\} = \left\{ \mathbf{x} \mid \mathbf{A_2^k} \mathbf{x} \leq \mathbf{q_2^k} \right\} = F_2^k$$

Basic Equivalent Transformations

$$V(\mathbf{P}, \mathbf{R}, \mathbf{A^k}, \mathbf{Q^k}) \xrightarrow[\text{A series of basic transformations}]{\text{NOVA}} V'(\mathbf{P}', \mathbf{R}', \mathbf{A}'^{\mathbf{k}}, \mathbf{Q}'^{\mathbf{k}})$$

Network view used in the example:

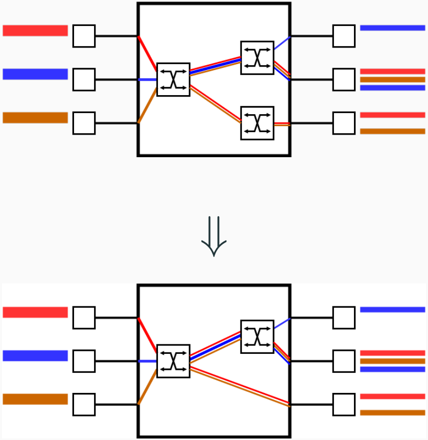
$\mathbf{R}(\mathbf{A^T})$	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
r_1	1	0	0	1	0	0	1	0
r_2	1	0	0	0	1	0	0	1
b_1	0	1	0	1	0	1	0	0
b_2	0	1	0	1	0	0	1	0
y_1	0	0	1	1	0	0	1	0
y_2	0	0	1	0	1	0	0	1
hop	1	1	1	1	1	1	1	1
bw	100	100	100	150	150	70	70	70



Basic Transformation Example: Aggregation

- Equivalence condition: Same row vector $\mathbf{R}_j^T$ & $\mathbf{A}_j^k$
- Merge columns with variant QoS metric algebra

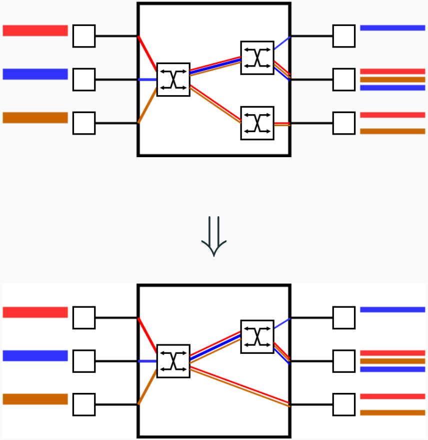
$\mathbf{R}(\mathbf{A}^\mathbf{T})$	e_1	e_2	e_3	e_4	e_5	e_6	e_7	e_8
r_1	1	0	0	1	0	0	1	0
r_2	1	0	0	0	1	0	0	1
b_1	0	1	0	1	0	1	0	0
b_2	0	1	0	1	0	0	1	0
y_1	0	0	1	1	0	0	1	0
y_2	0	0	1	0	1	0	0	1
hop	1	1	1	1	1	1	1	1
bw	100	100	100	150	150	70	70	70



Basic Transformation Example: Aggregation

- Equivalence condition: Same row vector $\mathbf{R}_j^T$ & $\mathbf{A}_j^k$
- Merge columns with variant QoS metric algebra

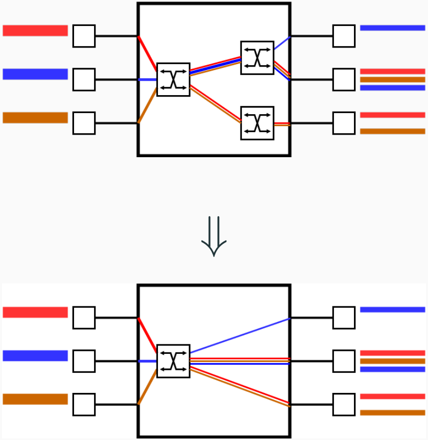
$\mathbf{R}(\mathbf{A}^\mathbf{T})$	e_1	e_2	e_3	e_4	e_6	e_7	e_8
r_1	1	0	0	1	0	1	0
r_2	1	0	0	0	0	0	1
b_1	0	1	0	1	1	0	0
b_2	0	1	0	1	0	1	0
y_1	0	0	1	1	0	1	0
y_2	0	0	1	0	0	0	1
hop	1	1	1	1	1	1	2
bw	100	100	100	150	70	70	70



Basic Transformation Example: Decomposition

- Equivalence condition: $\mathbf{A}_j^k \mathbf{x} \leq q_j^k$ is redundant and $\mathbf{R}_j = \sum_r \mathbf{R}_j^{(r)}$, $\mathbf{A}_j^k = \sum_r \mathbf{A}_j^{k(r)}$
- Split columns so that new columns can be aggregated

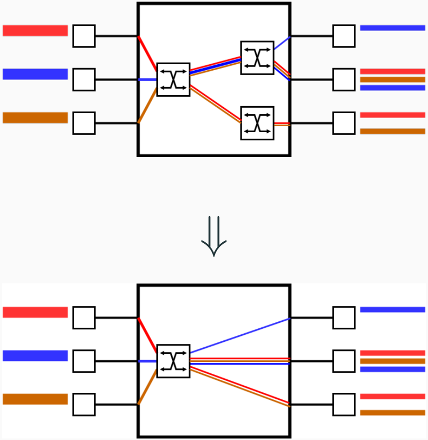
$\mathbf{R}(\mathbf{A}^\mathbf{T})$	e_1	e_2	e_3	e_4	e_6	e_7	e_8
r_1	1	0	0	1	0	1	0
r_2	1	0	0	0	0	0	1
b_1	0	1	0	1	1	0	0
b_2	0	1	0	1	0	1	0
y_1	0	0	1	1	0	1	0
y_2	0	0	1	0	0	0	1
hop	1	1	1	1	1	1	2
bw	100	100	100	150	70	70	70



Basic Transformation Example: Decomposition

- Equivalence condition: $\mathbf{A}_j^k \mathbf{x} \leq q_j^k$ is redundant and $\mathbf{R}_j = \sum_r \mathbf{R}_j^{(r)}$, $\mathbf{A}_j^k = \sum_r \mathbf{A}^{k}{}_j^{(r)}$
- Split columns so that new columns can be aggregated

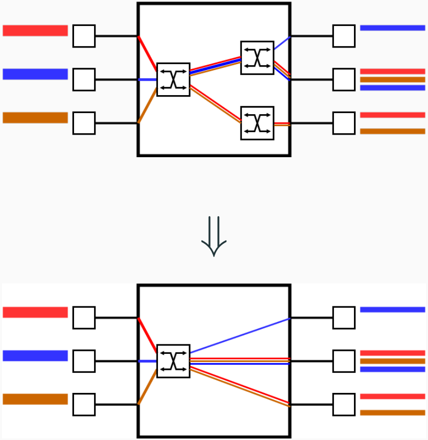
$\mathbf{R}(\mathbf{A}^\mathbf{T})$	e_1	e_2	e_3	e_4^1	e_4^2	e_6	e_7	e_8
r_1	1	0	0	0	1	0	1	0
r_2	1	0	0	0	0	0	0	1
b_1	0	1	0	1	0	1	0	0
b_2	0	1	0	0	1	0	1	0
y_1	0	0	1	0	1	0	1	0
y_2	0	0	1	0	0	0	0	1
hop	1	1	1	1	1	1	1	2
bw	100	100	100	150	150	70	70	70



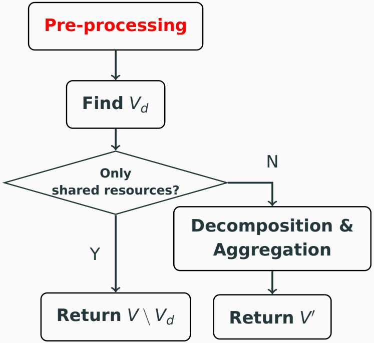
Basic Transformation Example: Decomposition

- Equivalence condition: $\mathbf{A}_j^k \mathbf{x} \leq q_j^k$ is redundant and $\mathbf{R}_j = \sum_r \mathbf{R}_j^{(r)}$, $\mathbf{A}_j^k = \sum_r \mathbf{A}_j^{k(r)}$
- Split columns so that new columns can be aggregated

$\mathbf{R}(\mathbf{A}^\mathbf{T})$	e_1	e_2	e_3	e_6	e_7	e_8
r_1	1	0	0	0	1	0
r_2	1	0	0	0	0	1
b_1	0	1	0	1	0	0
b_2	0	1	0	0	1	0
y_1	0	0	1	0	1	0
y_2	0	0	1	0	0	1
hop	1	1	1	2	2	2
bw	100	100	100	70	70	70



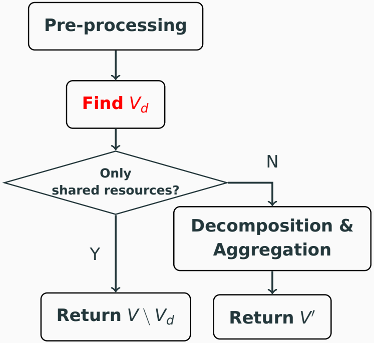
Overall Algorithm



Aggregate columns with the same row vector (R_j^T, A_j^k)

- A pre-processing step
- Avoid corner case in redundancy check (identical constraints)

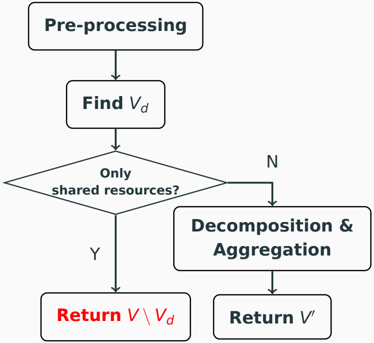
Overall Algorithm



Find decomposable columns V_d

- Equivalent to finding redundant linear constraints
- Well-studied problem

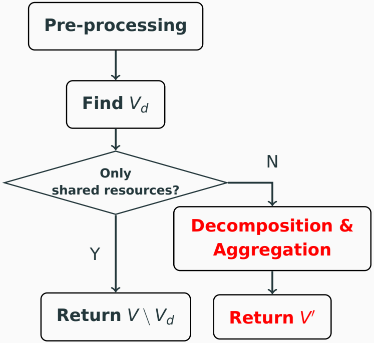
Overall Algorithm



Check if only shared resources are requested

- Equivalent to One-big switch abstraction if no bottlenecks are within the network

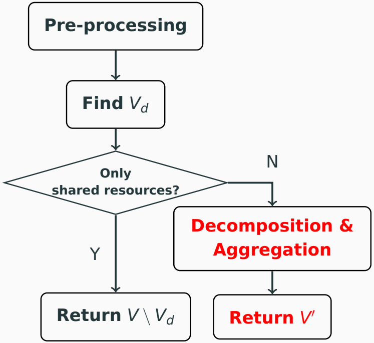
Overall Algorithm



Decompose $\forall v \in V_d$ with unit basis

- Bounded size $(|V| - |V_d| + |F|)$
- Low computation overhead $O(|V_d||F|)$
- Equivalent to end-to-end abstraction if $V = V_d$ (only end-to-end metrics are requested)

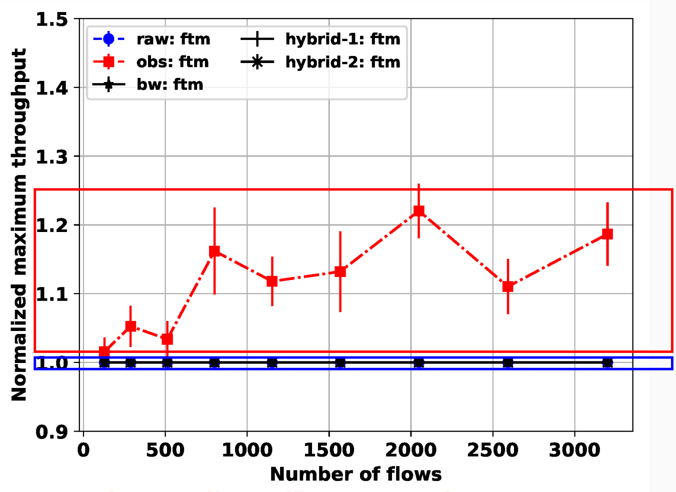
Overall Algorithm



Techniques to improve performance:

- Reduce space overhead: Aggregate the new columns immediately
- Leverage parallel processing: Use Map-Reduce-like divide-and-conquer

Evaluation: Verifying Equivalence

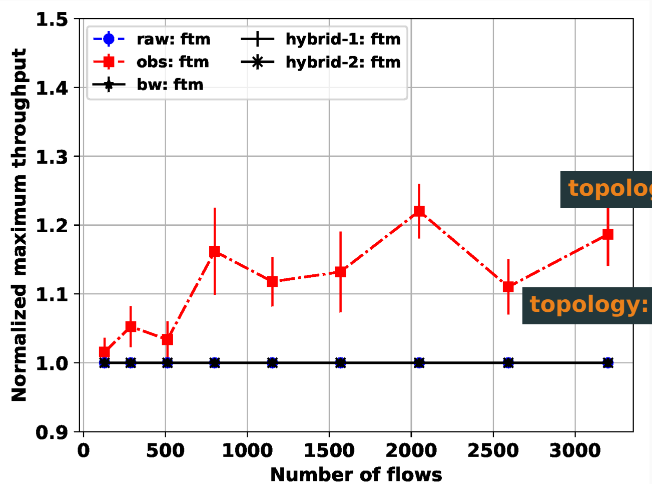


topology: Kdl traffic pattern: few-to-many

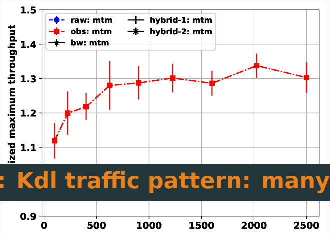
One-big switch:
infeasible solution

NOVA:
overlap with raw network view

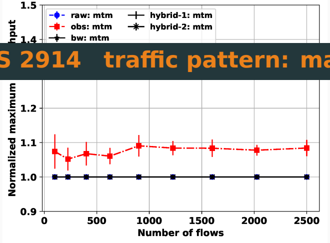
Evaluation: Verifying Equivalence



topology: Kdl   traffic pattern: few-to-many



topology: Kdl traffic pattern: many-to-many



topology: AS 2914   traffic pattern: many-to-ma

Evaluation: Effective Factors

- Topology
- Traffic pattern
- Number of flows
- Redundancy check algorithm

Evaluation: Effective Factors

- Topology
- Traffic pattern
- Number of flows
- Redundancy check algorithm

Determined by the nature of the network and the application

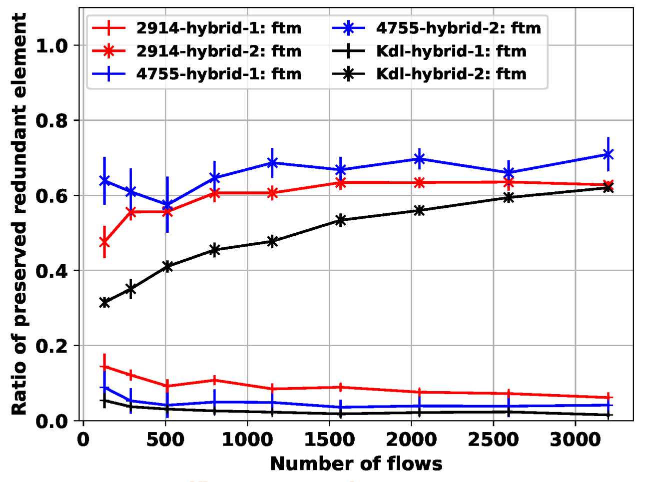
Evaluation: Effective Factors

- Topology
- Traffic pattern
- Number of flows
- Redundancy check algorithm

Can be controlled by NOVA

Evaluation: Topology

Normalized redundancy preservation

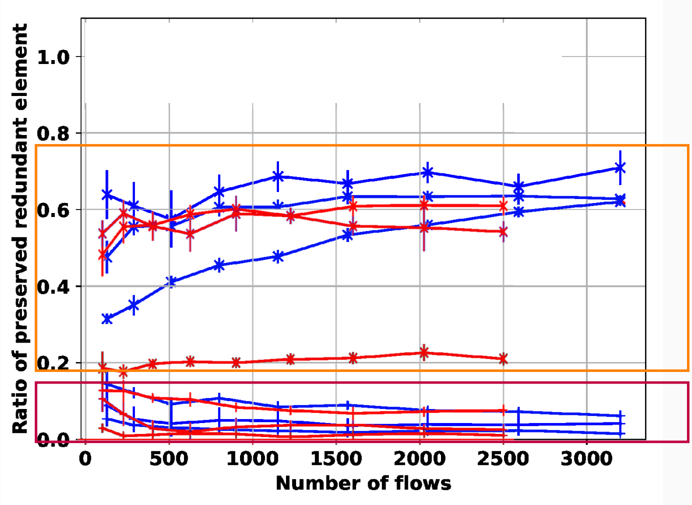


traffic pattern: few-to-many

Topology can effect the redundancy preservation

Evaluation: Traffic Pattern

Redundancy preservation

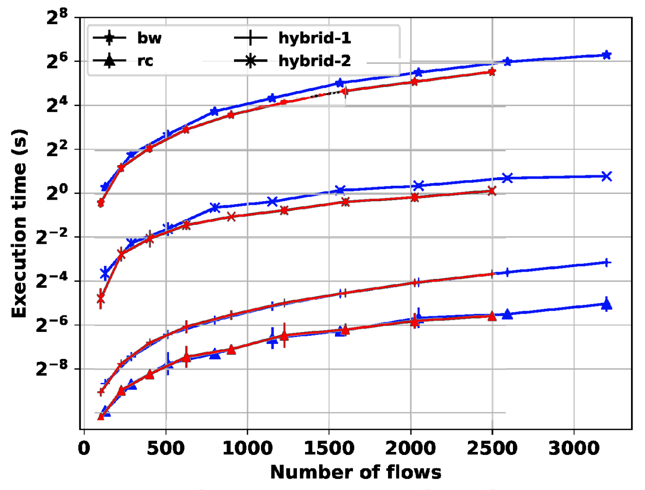


Relaxed redundancy check: **Many-to-many** **has better reduction ratio than** **few-to-many**

Strict redundancy check: **No significant difference**

Evaluation: Traffic Pattern

Computation overhead

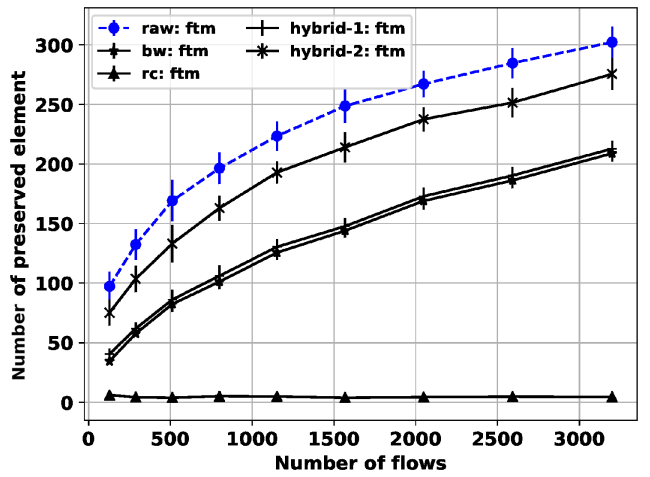


topology: AS 2914, 8 threads

Few-to-many and many-to-many patterns have no significant difference

Evaluation: Number of flows

Redundancy reduction

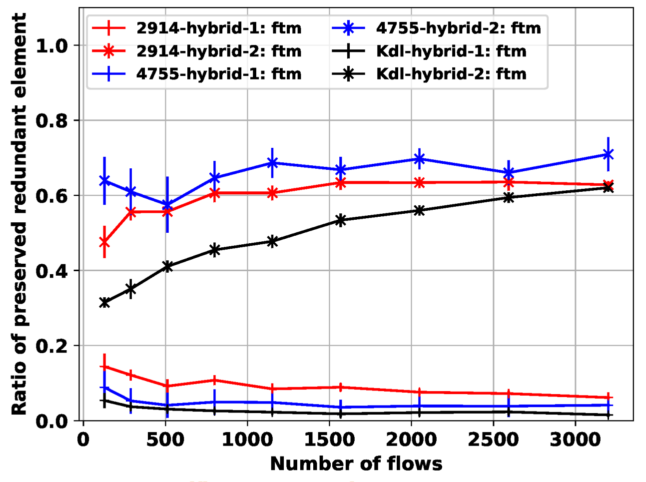


topology: AS 4755, traffic pattern: few-to-many

More flows,
more preserved elements
more information leaked

Evaluation: Number of flows

Redundancy reduction

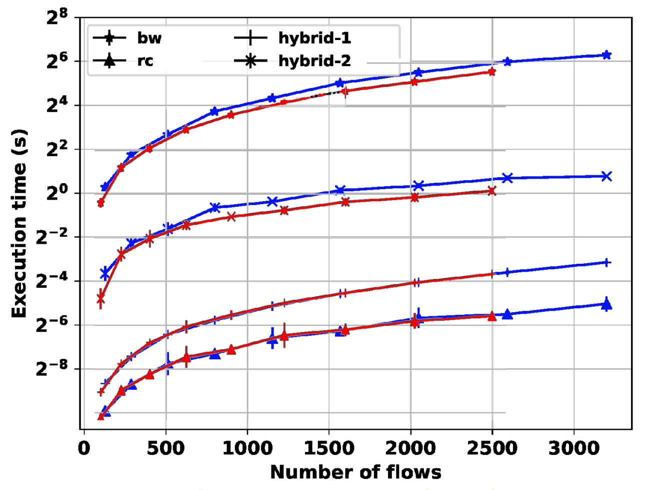


traffic pattern: few-to-many

Effect on reduction ratio depends on topology and redundancy check algorithm

Evaluation: Number of flows

Computation overhead

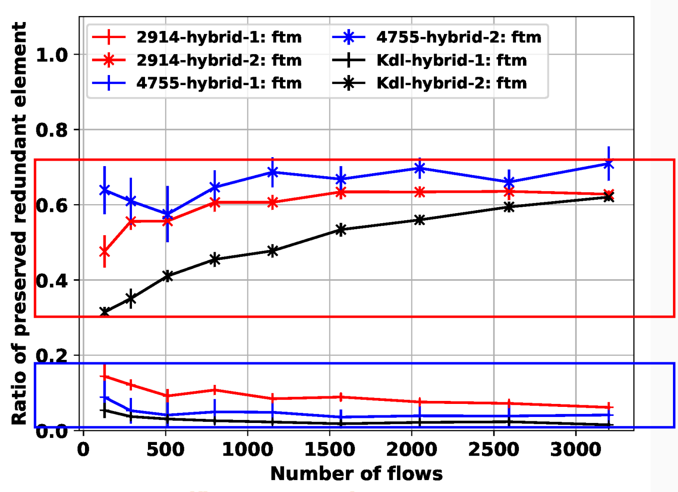


topology: AS 2914, 8 threads

**More flows,
larger computation time**

Evaluation: Redundancy Check Algorithm

Normalized redundancy preservation



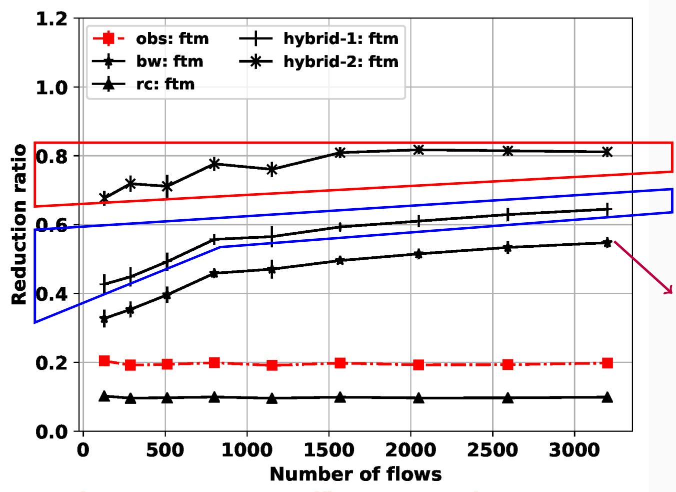
traffic pattern: few-to-many

Relaxed redundancy check: More remaining redundant elements

Strict redundancy check: Less remaining redundant elements

Evaluation: Redundancy Check Algorithm

Normalized communication overhead



Relaxed redundancy check:
Larger overhead (∼ 4x OBS)

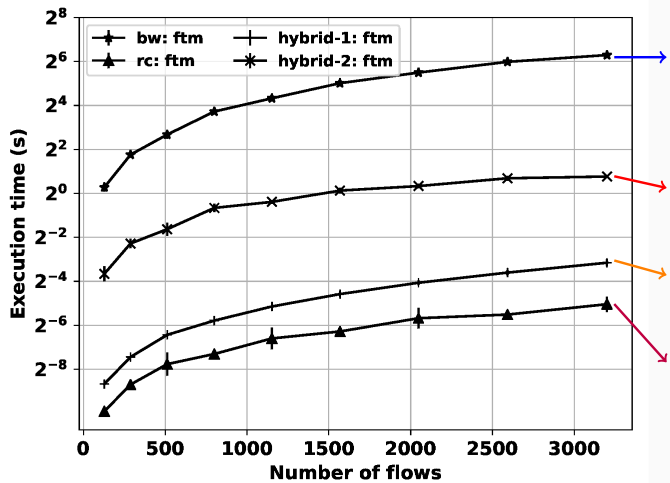
Strict redundancy check:
Smaller overhead (∼ 3x OBS)

Optimal size without information loss

topology: AS 2914, traffic pattern: few-to-many

Evaluation: Redundancy Check Algorithm

Computation overhead



Time of strict redundancy check algorithm (~ 5s for ~500 flows)

Time of relaxed redundancy check algorithm (<3s for ~3000 flows)

Time of decomposition & aggregation

Time to compute P × R (Routing cost only)

topology: AS 2914, traffic pattern: few-to-many, 8 threads

Evaluation: Effective Factors

- **Topology**
- **Traffic pattern**
- **Number of flows**
- **Redundancy check algorithm**

- Can effect the reduction ratio and computation time
- Usually does not change

Evaluation: Effective Factors

- **Topology**
- **Traffic pattern**
- **Number of flows**
- **Redundancy check algorithm**

- Mostly effect the relaxed redundancy check algorithm
- No significant correlations on computation time
- Usually does not change

Evaluation: Effective Factors

- **Topology**
- **Traffic pattern**
- **Number of flows**
- **Redundancy check algorithm**

- More flows, more information leaked, more communication overhead and computation time
- Mostly effect the relaxed redundancy check algorithm

Evaluation: Effective Factors

- **Topology**
- **Traffic pattern**
- **Number of flows**
- **Redundancy check algorithm**

- Strict redundancy check: Better privacy protection, less communication overhead, more computation overhead
- Relaxed redundancy check: More information leaked, more communication overhead, less computation overhead

Summary

Summary and Future Work

In this paper, we

- Identify the problem of providing on-demand network views and the limitations of existing works
- Extend the QoS metric algebra and model the Application-Network Collaborative Optimization
- Define equivalent network view and design novel and efficient algorithms to construct it

Conclusion

Evaluations show that:

- NOVA produces equivalent network view
- NOVA can effectively reduce redundant information and communication overhead with strict redundancy check algorithm
- NOVA can differentiate privacy by choosing different redundancy check algorithms and limiting the number of flows in a request

Future Work

- From a single user to multiple users
- From predetermined paths to application-aware path optimization
- From fixed reserved resource to dynamic resource allocation
- From simulated results to real running use cases

Thank you!

Q & A

QoS Metric Algebra

“Routing algebra” by Sobrinho[2]: $(P, S, w, \circ, \oplus, \preceq)$

- P: Set of paths
- S: Range of a metric
- $w : P \mapsto S$: Weight function
- $\circ : P \times P \mapsto P$: Concatenation operator
- $\oplus : S \times S \mapsto S$: Logical “plus” operator
- $\preceq: S \times S \mapsto \{0, 1\}$: Logical “compare” operator

---

[2] João Luís Sobrinho. “Algebra and algorithms for QoS path computation and hop-by-hop routing in the Internet”. In: *IEEE/ACM Transactions on Networking (TON)* 10.4 (2002). 00328, pp. 541–550.

QoS Metric Algebra

Variant metric algebra: $(P, S, w, \circ, \oplus, \preceq, \otimes)$

- P: Set of paths
- S: Range of a metric
- $w : P \mapsto S$: Weight function
- $\oplus : S \times S \mapsto S$: Logical “plus” operator
- $\otimes : \mathbb{R} \times S \mapsto S$: Logical “multiply” operator

QoS Metric Algebra Examples

Table 1: The Variant Routing Metric Algebra.

S	Weight function (w)	$w(p_1)$	$w(p_2)$	$w(p_1 \circ p_2)$	$N \otimes w(p_1)$	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