

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: January 7, 2016

K. Gao
Tsinghua University
X. Wang
Tongji University
Y. Yang
Yale University
G. Chen
Huawei
July 6, 2015

ALTO Extension: A Routing State Abstraction Service Using Declarative
Equivalence
draft-gao-alto-routing-state-abstraction-00.txt

Abstract

The Application-Layer Traffic Optimization (ALTO) protocol has defined multiple services (e.g., network maps, cost maps, filtered maps, the endpoint cost service, and the endpoint property service) to provide network state information to network applications. In a higher-level view, both the cost maps and the endpoint cost service can be considered as providing views into the routing state of a network (i.e., the path properties). A drawback of these existing services, however, is that they are static, application-oblivious views, without guidance from network applications. This document designs a new ALTO service named Routing State Abstraction using Declarative Equivalence (RSADE). Allowing applications to provide declarative guidance on the intended use of the network routing state, RSADE allows a network to compute compact, customized routing state abstraction beyond the existing services.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 7, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. The Multi-flow Scheduling Use Case	4
3. The RSADE Service	5
4. The RSADE Equivalence Condition	6
5. Security Considerations	8
6. IANA Considerations	9
7. Acknowledgments	9
8. References	9
8.1. Normative References	9
8.2. Informative References	9
Authors' Addresses	9

1. Introduction

The key services of the ALTO protocol [RFC7285] can be considered as information query services about the routing state of a network. Specifically, a cost map of an ALTO metric allows a network application to look up the end-to-end value of the given metric, for the routing path(s) from a given source to a given destination. The endpoint cost service provides a similar service.

The recent advance of newer network architectures such as SDN, however, reveals that the existing services may have limitations. First, the existing services distinguish routing state at the host

level. This is reasonable in a traditional network such as a network using destination IP based routing. The emergence of new techniques such as SDN using OpenFlow may convert more networks to use more fine-grained routing, such as the 5-tuple (source and destination IPs, source and destination ports, and protocol) routing. In such a setting, revealing routing state (e.g., cost) at the granularity of endhosts may be too coarse. For example, for a network where port 80 HTTP traffic is routed differently from port 22 traffic, the existing services cannot provide the differentiation.

Second, the existing (routing state query) ALTO services are designed for relatively simple network applications. More complex network applications, such as the multi-flow scheduling application [I-D.yang-alto-path-vector], may need more complex routing state information for better application-level coordination. Let f be the network application (or network component) and let $\text{view}()$ be the function that constructs an abstract routing state view for f . One can see that $\text{view}()$ may compute an on-demand, instead of static, view that will depend on f . The existing ALTO services do not provide this customization capability.

A possibility to address the customization problem is that the network provides raw, complete routing state view. However, providing abstract views on top of raw network state, as ALTO does, can provide substantial benefits to both the network, which manages the network state, and the network applications, which consume the network state. First, a more compact abstract network state view can reduce the requirement on client scaling. The raw network state of a large network may consist of a large number of network devices. A consumer of such a large amount of information must be scalable. Second, an abstract network state view can better protect the privacy of the provider of the network. Third, an abstract network state view may substantially reduce the load of information updates.

The objective of this document is to design an ALTO extension service named Routing State Abstraction using Declarative Equivalence (RSADE) to address the preceding two issues. Specifically, RSADE provides a simple, declarative API for a network application to specify its need (i.e., requirements) of routing and topology state, and the network computes a minimal, but equivalent routing state to the network application. For simplicity, this document focuses on extending the endpoint cost service, leaving the aggregation aspects of using network aggregation maps as future work.

The organization of this document is organized as follows. Section 2 replicates the multi-flow scheduling example from [I-D.yang-alto-path-vector]. Section 3 gives an overview of the service, and Section 4 gives more details on specifying state

equivalence. Sections 5 and 6 discuss security and IANA considerations.

2. The Multi-flow Scheduling Use Case

A foundation of the ALTO services is the routing cost value (for a given metric) for each pair of source and destination. Although simple, this foundation may not convey enough information to some applications. This document uses a simple use case in [I-D.yang-alto-path-vector] to illustrate the issue. See [I-D.lee-alto-app-net-info-exchange] for earlier, more comprehensive discussions.

Consider a network as shown in Figure 1. The network has 7 switches (sw1 to sw7) forming a dumb-bell topology. Switches sw1/sw3 provide access on one side, sw2/sw4 provide access on the other side, and sw5-sw7 form the backbone. Endhosts eh1 to eh4 are connected to access switches sw1 to sw4 respectively. Assume that the bandwidth of each link is 100 Mbps. Assume that the network is abstracted with 4 PIDs, with each representing the hosts at one access switch.

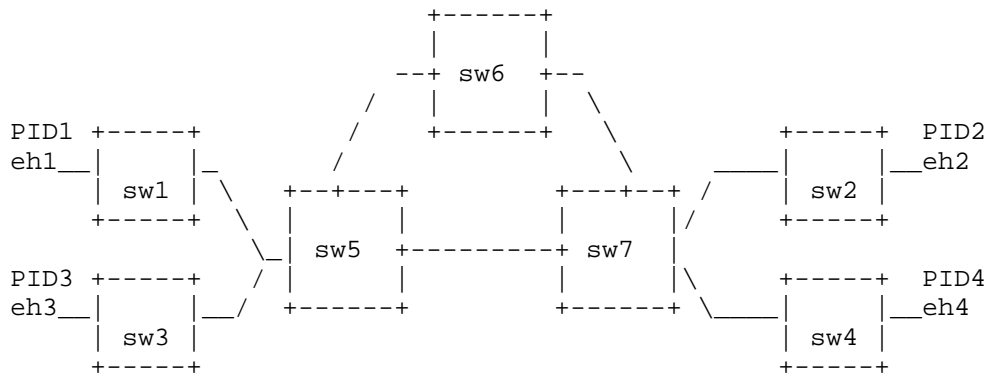


Figure 1: Raw Network Topology.

Consider an application overlay (e.g., a large data transfer system) which needs to schedule the traffic among a set of endhost source-destination pairs, say eh1 -> eh2, and eh3 -> eh4. The application can request a cost map (or endpoint cost service) providing end-to-end available bandwidth, using 'available bw' as cost-metric and 'numerical' as cost-mode, where the 'available bw' between two endhosts represents their available bandwidth, if no other applications use shared resources.

Assume that the application receives from the cost map that both eh1 -> eh2 and eh3 -> eh4 have bandwidth 100 Mbps. It cannot determine that if it schedules the two flows together, whether it will obtain a total of 100 Mbps or 200 Mbps. This depends on whether the routing of the two flows shares a bottleneck in the underlying topology:

- o Case 1: If the two flows use different paths in the current routing state, for example, when the first uses sw1 -> sw5 -> sw7 -> sw2, and the second uses sw3 -> sw5 -> sw6 -> sw7 -> sw4. Then the application will obtain 200 Mbps.
- o Case 2: If the two flows share a bottleneck in the current routing state, for example, when both use the direct link sw5 -> sw7, then the application will obtain only 100 Mbps.

To allow applications to distinguish the two aforementioned cases, the network needs to provide more details on the routing state. A naive solution to this problem, then, is to return the two complete, detailed routes and the available bandwidth of each link on the routes. But this may not be desirable, as the application may not need the details and/or may not have the permission to see networks details.

Now consider what route abstraction can achieve. Assuming case 2 (shared bottleneck), it is sufficient for the network to return a single abstract link for each flow: `anel(100Mbps)`, where `ane` stands for abstract network element, and the number in the number 100Mbps denotes its capacity.

Consider a variation of the preceding case. Assume that the capacity of the link from sw1 to sw5 is 70 Mbps, while the rest are still at 100 Mbps. Then the abstract route from eh2 to eh4 becomes `anel(100Mbps)` and `ane2(70Mbps)`.

3. The RSADE Service

The more the network knows about what a network application `f` needs regarding a routing state query, the more concise the network response can be. Hence, an extreme API is that the complete network application `f` (i.e., the code and related state) is sent to the network. This, however, can create substantial complexity in the routing-state query component, as even some simple program properties (e.g., halting) are already difficult to analyze. Also, in settings such as interdomain, the owner of the function `f` may not want to provide the complete `f` to the network.

Another extreme API is that each routing state query provides only the most basic information (i.e., the source and the destination).

This, however, does not provide enough information for the routing-state service to compute efficient route abstraction/compression. Hence, the returned routes will be independent of individual functions, missing out opportunities on abstraction or compression.

The RSADE service tries to strike a balance between the two extremes. Figure 1 gives the grammar to specify the query information that a network application sends to the network:

```
rs-query := flow-list equiv-cond
flow-list := flow [flow-list]
flow := generic-match-condition
```

Specifically, the first component of a RSADE query is a list of flows (flow-list). Each flow in the list is specified by a match condition, as in OpenFlow.

The second component of the query input is the declared equivalence condition. A particular type of equivalence condition, in the context of routing-state query, is the equal range condition. We give the detailed specification of the condition in Section 4.

After receiving an RSADE request, the network retrieves the route for each flow, and then computes the result after compression (abstraction). RSADE may allow a network application to specify an indicator, on whether it wants to receive incremental updates to the query results, achieving push notification. The push notification is implemented using HTTP SSE [Roome-SSE].

4. The RSADE Equivalence Condition

Let attr (e.g., delay) be a vector for a given link attribute. Let vector $R[i]$ represent the result of route lookup for flow i , where $R[i][e]$ is the fraction of traffic of flow i on link e , according to the current routing state. For example, the result of route lookup for the use case in Section 2 can be represented as the following:

	R[0]	R[1]	avabw	delay	bg-tr
link1	1	0	1G	2ms	...
link2	1	0	100M	5ms	...
link3	1	1	100M	5ms	
Link4	1	1	100M	5ms	
link5	0	1	100M	7ms	
link6	0	1	1G	4ms	
...					
linkM					

Although a routing-state query without abstraction/compression will return all of the data shown above, route abstraction/compression will select only a subset link attributes (columns) and some links (rows). Elimination of links from the complete result achieves compression but may result in loss of information to the application. Hence, a specification on conditions whether the elimination of a set of links from the complete result leads to information loss or not is the key to the problem definition. Such a specification, however, can be provided only by the application itself.

Specifically, in the general case, the result from the routing-state query will become the input parameters for the algorithms in the network application, to help the application to make decisions. Let x be the vector of the decision variables in the application. Then, one can identify that a generic structure of the application is to solve/optimize $\text{obj}(x)$, subject to two types of constraints on x : (1) those do not involve the results from the routing state query; and (2) those do. Let the first type limit x in X_0 . Consider the second type. The state of art in algorithmic design typically handles only linear constraints, and hence the set S of constraints of this type will be of the format $a_k x \leq b_k$, where a_k is a vector, and b_k a constant. Hence, it is in a_k or b_k where the result from the routing-state query appears. Let $A x \leq b$ as a matrix format to represent the whole set of constraints.

Now, consider the case that a link appears in the complete result of a RSADE query, but its parameters do not appear in a boundary constraint among the aforementioned constraints, then the link may not need to appear in a compressed RSADE query.

[Equivalence]: Two constraint sets S_1 and S_2 of a network function are equivalent if and only if they limit the decision variables in the same way: $X_0 \wedge \{x: A_1 x \leq b_1\} = X_0 \wedge \{x: A_2 x \leq b_2\}$.

[Redundant]: A constraint s is redundant to a constraint set S if and only if s in S and the two sets S and $S - \{s\}$ are equivalent.

[Minimal Constraint Set]: A constraint set S is minimal if and only if for any s in S , s is not redundant.

[Equivalent Routing-State Query]

A declarative equivalence based routing-state query is one where the querier (application) declares X_0 and a set of constraints $S = \{a_k x \leq b_k\}$. If the attribute of a link does not appear in a minimal constraint set, the link can be eliminated from the routing-state result.

A concern one may have is that the preceding definition may be limited. Consider the case of hierarchical networks, where the upper-layer network (i.e., the network application) conducts routing (traffic engineering) in its layer and uses RSADE to obtain the state of the lower layer. Let flows be the $n(n-1)$ source-destination pairs in the upper layer network with n nodes. Let x be the set of decision variables controlling the routing in the upper-layer, where each element is the routing on each of the preceding flows. Let X_0 encode the constraints on traffic demand. We have the following result:

[UTE Completeness]

Any upper-layer routing (traffic engineering) algorithm where the goal of RSADE in the lower-layer network is to avoid congestion of shared links or shared risk groups can be implemented using the declarative equivalence based routing-state query. We refer to this as the upper-layer traffic engineering (UTE). Let $A = R$ and $b = \text{cap}$. Then the RSADE query returns a link only if the link may become a bottleneck in the upper layer network.

5. Security Considerations

This document has not conducted its security analysis.

6. IANA Considerations

This document requires the definition of a new cost-mode named path-vector.

7. Acknowledgments

The author thanks discussions with Jun Bi and Andreas Voellmy.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

8.2. Informative References

- [I-D.lee-alto-app-net-info-exchange]
Lee, Y., Bernstein, G., Choi, T., and D. Dhody, "ALTO Extensions to Support Application and Network Resource Information Exchange for High Bandwidth Applications", draft-lee-alto-app-net-info-exchange-02 (work in progress), July 2013.
- [I-D.yang-alto-path-vector]
Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Topology Extension: Path Vector as a Cost Mode", draft-yang-alto-path-vector-00 (work in progress), March 2015.
- [RFC7285] Alimi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.

Authors' Addresses

Kai Gao
Tsinghua University
30 Shuangqinglu Street
Beijing 100084
China

Xing (Tony) Wang
Tongji University
4800 CaoAn Road
Shanghai 210000

Y. Richard Yang
Yale University
51 Prospect St
New Haven CT
USA

Email: yry@cs.yale.edu

G. Robert Chen
Huawei
Nanjing
China

Email: chenguohai@huawei.com