Supporting Bottleneck Structure Graphs in ALTO: Use Cases and Requirements
draft-giraltyellamraju-alto-bsg-requirements-01

Abstract

This document proposes an extension to the base Application-Layer Traffic Optimization (ALTO) protocol to support bottleneck structures as an efficient representation of the state of a network. Bottleneck structures are efficient computational graphs that allow network operators and application service providers to optimize application performance in a variety of communication problems including routing, flow control, flow scheduling, bandwidth prediction, and network slicing, among others. This document introduces a new abstraction called Bottleneck Structure Graph (BSG) and the necessary requirements to integrate it into the ALTO standard.

About This Document

This note is to be removed before publishing as an RFC.

The latest revision of this draft can be found at https://giralt.github.io/draft-ietf-alto-gradient-graph/draft-giraltyellamraju-alto-bsg-requirements.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-giraltyellamraju-alto-bsg-requirements/.

Discussion of this document takes place on the WG Working Group mailing list (mailto:alto@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/alto/.

Source for this draft and an issue tracker can be found at https://github.com/giralt/draft-ietf-alto-gradient-graph.
Internet-Draft  Supporting Bottleneck Structure Graphs i      March 2022

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 https://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 23 September 2022.

Copyright Notice

Copyright (c) 2022 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
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 Revised BSD License text as
described in Section 4.e of the Trust Legal Provisions and are
provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction ............................................. 3
2. Conventions and Definitions ............................. 4
3. Brief Introduction to Bottleneck Structures .......... 4
   3.1. Example of Bottleneck Structure .................. 4
   3.2. Propagation Properties ............................ 7
   3.3. Forces of Interaction among Flows and Links ........ 7
   3.4. Ripple Effects in a Communication Network ....... 8
   3.5. Not all Bottleneck Links Are Born Equal .......... 8
   3.6. Quantifying the Ripple Effects .................. 9
   3.7. Types of Bottleneck Structures .................. 11
   3.8. Computing Optimized Network Reconfigurations .... 12
   3.9. Types of Network Reconfigurations ............... 12
4. ALTO Bottleneck Structure Service Use Cases .......... 15
   4.1. Application Rate Limiting for CDN and Edge Cloud
        Applications ....................................... 15

Ros-Giralt, et al.      Expires 23 September 2022
1. Introduction

Bottleneck structures have been recently introduced in [G2-SIGCOMM] and [G2-SIGMETRICS] as efficient computational graphs that embed information about the topology, routing and flow information of a network. These computational graphs allow network operators and application service providers to compute network derivatives that can be used to make traffic optimization decisions. For instance, using the bottleneck structure of a network, a real-time communication (RTC) application can efficiently infer the multi-hop end-to-end available bandwidth, and use that information to tune the encoder’s transmission rate and optimize the user’s Quality of Experience (QoE). Bottleneck structures can be used by the application to address a wide variety of communication optimization problems, including routing, flow control, flow scheduling, bandwidth prediction, and network slicing, among others.
This document introduces a new abstraction called Bottleneck Structure Graph (BSG) and the necessary requirements to integrate it into the existing ALTO services (Network Map, Cost Map, Entity Property Map and Endpoint Cost Map) exposing the properties of the bottleneck structure to help optimize application performance. Use cases are also introduced to motivate the relevancy of bottleneck structures in the context of the ALTO standard and support the description of the integration requirements.

2. Conventions and Definitions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Brief Introduction to Bottleneck Structures

[G2-SIGMETRICS] and [G2-SIGCOMM] introduce a new mathematical framework to optimize network performance called the Quantitative Theory of Bottleneck Structures (QTBS). The core building block of QTBS is a computational graph called _bottleneck structure_, which allows to qualify and quantify the forces of interactions that flows and bottleneck links exert on each other. QTBS builds the bottleneck structure by assuming that flows in a network aim at maximizing throughput while ensuring fairness. This general principle holds for all the relevant congestion control algorithms used in IP networks (e.g., TCP Cubic, BBR, QUIC, etc.).

3.1. Example of Bottleneck Structure

Consider as an example the following network configuration:
Figure 1: Network configuration example.

Each link $l_i$ is represented by a squared box and has a capacity $c_i$. For instance, link $l_1$ is represented by the top most squared box and has a capacity of $c_1=25$ units of bandwidth. In addition, each flow is represented by a line that passes through the set of links it traverses. For instance, flow $f_6$ traverses links $l_1$, $l_2$ and $l_3$. 
The bottleneck structure of this network corresponds to the following digraph (see [G2-TREP] for details on how a bottleneck structure is computed):

```
+-----+  +-----+               +-----+
|      |  |      |               |      |
|  f1  <-->  l1  <--------------->  f6  |
|      |  |      |  +------------+      |
+------+  +--^---+  |            +---+--+
|      |                |
|      |                |
+--v---+  |                |
|      |  |                |
|  f3  |  |                |
|      |  |                |
+--+---+  |                |
|      |                |
+-----+  +--|---+  |                |
|      |     |      |  +------+
|  l2  <----|  f4  +--->  l3  |  |  l4  |
|      |     |      |   |      |  |      |
+--^---+     +------> +--^---+  +---^--+
|                       |          |
+--v---+                   | +------+ |
|      |                   | |      | |
|  f2  |                   +->  f5  <-+ |
|      |                     |      |
+------+                     +------+
```

**Figure 2:** Bottleneck structure of the network in Figure 1.

The bottleneck structure is interpreted as follows:

* Links and flows are represented by vertices in the graph.

* There is a directed edge from a link l to a flow f if and only if flow f is bottlenecked at link l.

* There is a directed edge from a flow f to a link l if and only if flow f traverses link l.

For instance, in Figure 2 we have that flow f3 is bottlenecked at link l1 (since there is a directed edge from l1 to f3) and it traverses links l1 and l2 (since there is a directed edge from f3 to l1 and from f3 to l2). Note that when a flow is bottlenecked at a link, then the edge connecting them in the bottleneck structure must necessarily be bidirectional. This is because a flow that is
bottlenecked at a link, must necessarily traverse that link too. Indeed, in Figure 2 we can see that all the directed edges from a link to a flow, are in fact bidirectional edges. This is important to ensure that bottleneck structures correctly model how perturbations in a network propagate, as we explain in the next section.

3.2. Propagation Properties

Under the assumption of max-min fairness [GALLAGER], QTBS demonstrates the following two properties [G2-SIGMETRICS]:

* Property 1. Flow perturbation*. An infinitesimal change in the transmission rate of a flow $f$ will have an effect on the transmission rate of a flow $f'$ if and only if the bottleneck structure has a directed path from flow $f$ to flow $f'$.

* Property 2. Link perturbation*. An infinitesimal change in the capacity of a link $l$ will have an effect on the transmission rate of a flow $f'$ if and only if the bottleneck structure has a directed path from link $l$ to flow $f$.

The above two properties qualitatively relate to the classic question in chaos theory: Can the flap of a butterfly’s wings in Brazil set off a tornado in Texas? [LORENZ] Obviously a butterfly alone cannot create a tornado, but every element is interconnected in a distributed system, and even the flap of a butterfly’s wings in Brazil will have an effect in Texas. Bottleneck structures are graphs that characterize and quantify such type of effects in a communication network. In particular, a bottleneck structure reveals how a perturbation propagates through the network, describing which flows will be affected and by what magnitude.

3.3. Forces of Interaction among Flows and Links

Bottleneck structures are powerful computational graphs because they are able to capture the forces of interaction that flows and bottleneck links exert on each other. These forces of interaction are in general non-intuitive, even for a small simple network configuration like the one in Figure 1. For instance, from Property 2, the bottleneck structure reveals that a small variation in the capacity of link 12 (e.g., in a wireless network, a variation in the capacity of a link could be due to a change in the signal to noise ratio of the communication channel) will propagate through the network and have an impact on the transmission rate of flows $f_2$, $f_4$ and $f_5$ (since from Property 2, the bottleneck structure has a directed path from link 12 to each of these flows). However, such a perturbation will have no effect on the transmission rate of flows...
f1, f3 and f6 (since there is no path from l2 to any of these other flows). Similarly, from property 1, a small perturbation on the rate of flow f4 (e.g., this could be due to the effect of a traffic shaper altering the transmission rate of flow f4), will have an impact on the rate of flows f2 and f5, but it will have no effect on the rate of flows f1, f3 and f6.

3.4. Ripple Effects in a Communication Network

As another example, given the network in Figure 1, it is also not intuitive to foresee that flows f1 and f5 are related to each other by the forces of interaction inherent to the communication network, even though they do not traverse any common link. Specifically, flow f1 traverses link l1, while flow f5 traverses links l3 and l4. In between both flows, there is an additional hop (link l2) further separating them. Despite not being directly connected, the bottleneck structure reveals that a small perturbation on the performance of flow f1 (i.e., a change in its transmission rate), will create a ripple effect that will reach flow f5, affecting its transmission rate. In particular, the perturbation on flow f1 will propagate through the bottleneck structure and reach flow f5 via the following two paths:

f1 -> l1 -> f3 -> l2 -> f4 -> l3 -> f5
f1 -> l1 -> f6 -> l3 -> f5

It is also not intuitive to see that the reverse is not true. That is, a small perturbation on flow f5, will have no effect on flow f1, since the bottleneck structure has no direct path from vertex f5 to vertex f1. In [G2-SIGMETRICS], extensive empirical validation of these results is presented for a variety congestion-controlled IP networks.

3.5. Not all Bottleneck Links Are Born Equal

Bottleneck structures also reveal that not all bottleneck links have the same relevancy. In Figure 2, links at the top of the graph have a higher impact on the overall performance of the network than links at the bottom. For instance, consider link l1. A variation on its capacity will create a ripple effect that will impact the performance of all the flows in the network, since all flow vertices are reachable from vertex l1 according to the bottleneck structure. In contrast, link l3 has a much smaller impact on the overall performance of the network, since a variation of its capacity will affect flow f5, but have no impact on any of the other flows. This information is valuable to network operators and application service providers as it can be used to make informed network optimization...
decisions. For instance, in edge computing, an operator could choose to place a containerized service (e.g., for extended reality, XR) on compute nodes that would yield communication paths traversing bottleneck links with lower impact on the overall performance of the network (See the use case in Section 4.5 for more details). Similarly, in network slicing (or capacity planning in general), operators could choose to allocate more bandwidth on those links that are more influential (i.e., those links that are at lower levels in the bottleneck structure) according to the expected traffic pattern in the network slice.

Overall, bottleneck structures provide a mechanism to rank bottleneck links according to their impact on the overall performance of the network. This information can be used in a variety of optimization problems, such as traffic engineering, routing, capacity planning, or resilience analysis, among others.

3.6. Quantifying the Ripple Effects

Bottleneck structures not only allow network operators to reason about the qualitative nature of the forces that flows and links exert on each other, but they also provide a mathematical framework to quantify such forces. In particular, the Quantitative Theory of Bottleneck Structures (QTBS) introduced in [G2-TREP] provides a mathematical framework that uses bottleneck structures as efficient computational graphs to quantify the impact that perturbations in a network have on all of its flows.

One of the core building blocks of the QTBS framework is the concept of _link and flow equations_, which mathematically characterize how a perturbation in a network propagates through each of the link and flow vertices in the bottleneck structure. (See [G2-TREP] for an exact mathematical formulation.) Because quantifying the effect of a perturbation on a system is nothing more than computing a derivative of the system’s performance with respect to the parameter that’s been perturbed, bottleneck structures can be used as efficient and scalable computational graphs to calculate flow and link derivatives in a communication network.

Consider for instance the computation of the following derivative:

\[ \frac{dF()}{dri} \]

where \( F() \) represents the total throughput of the network (the sum of all flows’ throughput) and \( ri \) is the transmission rate of flow \( fi \). For example, this expression can be used to compute the effect of traffic shaping a flow (slightly reducing its rate) on the total throughput of the network. Computing this derivative using a
traditional calculus approach is both very complex and costly, since it requires modeling the congestion control algorithm in the function \( F() \), for which there is no closed form solution. Using bottleneck structures, however, the computation of this derivative is both simple and inexpensive. It is simple because it can be done by applying an infinitesimal change to the rate of flow \( f_i \) and then using the link and flow equations to measure how this perturbation propagates through the bottleneck structure [G2-TREP], [G2-SIGCOMM]. It is also very efficient because the computation is performed by applying delta calculations on the bottleneck structure, without involving links and flows that are not affected by the perturbation. For instance, in Figure 1, the computation of \( \frac{dF()}{df_4} \) only requires recomputing the transmission rates of flows \( f_2 \) and \( f_5 \), without the need to recompute the rates of \( f_1 \), \( f_3 \) and \( f_6 \), since these other flows are not affected by the perturbation. In practice, QTBS provides a methodology to compute network derivatives two or three orders of magnitude faster than general purpose methods such as linear programming [G2-SC].

We finish this brief introduction to QTBS by stating the monotonic bandwidth allocation property that all bottleneck structures satisfy:

* **Property 3. Monotonic bandwidth allocation (MBA).** Let \( s_i \) be the transmission rate of the flows bottlenecked at link \( l_i \). Then, for any path in the bottleneck structure of the form

\[
l_1 \rightarrow f_1 \rightarrow l_2 \rightarrow f_2 \rightarrow (...) \rightarrow l_n \rightarrow f_n
\]

we have that

\[
s_1 < s_2 < (...) < s_n
\]

The MBA property is relevant in that it states that bottlenecks located at higher levels in the bottleneck structure will have more bandwidth available than those located at lower levels. For instance, this property indicates that an application requiring high bandwidth should route its traffic through paths that involve links at higher levels in the bottleneck structure. We will be using the MBA property to reason about application performance in some of the examples described in this document.
3.7. Types of Bottleneck Structures

While QTBS introduces a core definition of bottleneck structure (see Section 3.1), there exist multiple types of bottleneck structures that can be computed depending on the level of granularity and information desired by the operator. Next, we introduce three types of bottleneck structures that will be used in this document and that are suitable to optimize application performance in the context of the ALTO standard:

* **Flow gradient graph (FGG)**. This type of bottleneck structure corresponds to the base definition introduced in Section 3.1. The FGG has the finest level of granularity, including a vertex in each graph for each link and flow in the network. Therefore, an FGG can be relatively large (e.g., with millions of vertices).

* **Path gradient graph (PGG)**. One technique to reduce the size of the bottleneck structure without affecting its accuracy is to collapse all the vertices of the flows that follow the same path into a single vertex called a _path vertex_. The resulting bottleneck structure is called the path gradient graph (PGG). A PGG usually has 2 or 3 orders of magnitude less vertices than the FGG.

* **QoS-Path gradient graph (Q-PGG)**. Some networks assign different types of traffic to different QoS classes. A Q-PGG can model QoS by collapsing all the vertices of the flows that follow the same path and have the same QoS class into a single vertex called a _Q-path vertex_. A Q-PGG is slightly larger than a PGG (with about $|Q|$ times more vertices, where $|Q|$ is the number of QoS classes supported by the network) but still significantly smaller than the FGG.

For most of the applications, it is recommended to use a PGG, or a Q-PGG if the network supports QoS classes, since these bottleneck structures are significantly smaller and faster to process than an FGG, and it is often the case that the operator does not need to know flow-level information in order to make proper application performance optimization decisions. Note also that the PGG and the Q-PGG provide the additional security advantage of hiding flow-level information from the graph. This can be important to operators that are sensitive about security and privacy.
3.8. Computing Optimized Network Reconfigurations

A central element to the theory of bottleneck structures is the ability to efficiently compute derivatives on a network. Derivatives are a core building block of the optimization framework, as they reveal the directions (gradients) in the feasible set that can help bring the network to a higher level of performance. In this document, we will refer to these directions in the feasible set as _network reconfigurations_, since that’s what they effectively are in the physical world.

For instance, an example of network reconfiguration can be the action of rate limiting a flow carrying XR traffic to match the available bandwidth along its path with the goal to improve its QoE. Another example of network reconfiguration is the action of rerouting traffic through a new path in order to accelerate the transfer of a large backup data set between two cloud data centers. A third example can be the deployment of a new network slice in a 5G network in order to ensure the QoS of a V2X service. In each of these actions, the network configuration is moved along a direction (a gradient, if the change maximally improves the performance objective) within the feasible set of possible configurations.

While derivatives describe how the performance of a network changes when a very small (infinitesimal) change is applied to its configuration, network reconfigurations can accept changes to the network that are arbitrarily large. For instance, traffic shaping a set of flows to reduce their rates by 10 Mbps is a network reconfiguration that is not infinitesimal. We note that bottleneck structures can also be used to compute optimized network reconfigurations consisting of non-infinitesimal changes in the network. This can be done by first computing derivatives using the bottleneck structure to find a direction (gradient) in the feasible set, and then reconfiguring the network by following that direction. This process can be repeated iteratively until a final optimized reconfiguration is achieved. (See for example [G2-SIGCOMM] and [G2-TREP] for examples of algorithms using this technique.)

In the next section, we summarize some of the network reconfigurations that can be optimized by using bottleneck structures.

3.9. Types of Network Reconfigurations

The following is a list of some of the network reconfigurations that can be efficiently computed and optimized using bottleneck structures:
* *Flow routing*. Both the operation of routing a new flow or rerouting an existing flow on a network can be modeled as a perturbation, whose impact can be efficiently measured using bottleneck structures. In particular, QTBS can be used to resolve the joint congestion control and routing optimization problem for individual flows (see Section 3.1 in [G2-TREP]).

* *Traffic shaping*. Traffic shaping a flow corresponds to the action of taking a derivative with respect to the rate of the flow. Bottleneck structures can be used by network operators and application service providers to compute such perturbations. For instance, to accelerate a large scale data transfer, an application can use bottleneck structures to identify optimal traffic shaping configurations (see Section 3.3 in [G2-TREP]).

* *Bandwidth enforcement*. In high-performance networks that target close to 100% link utilization such as Google’s B4 network [B4-SIGCOMM], a centralized SDN controller is used collect the state of the network and compute an optimized multipath bandwidth allocation vector. The solution is then deployed at the edge of the network using a technique known as bandwidth enforcement [BE-SIGCOMM]. By using bottleneck structures to efficiently compute changes in the bandwidth allocated to each flow path, operators can efficiently derive improved bandwidth allocation vectors.

* *Flow scheduling*. When a flow initiates transmitting data on a network, it uses bandwidth along its path, creating a ripple effect that impacts the performance of other flows in the network. Similarly, the termination of a flow frees bandwidth along its path, creating another perturbation that propagates through the network. Bottleneck structures can efficiently model and compute the effect of flow arrival and departure in a communication network by using simple delta calculations according to the link and flow equations (see Section 3.6 and [G2-SIGCOMM]). This information can be used by applications that need to perform bulk data transfer to decide when to schedule a flow. More in particular, it can be used to enhance the ALTO Cost Calendar service [RFC8896].
* **Service placement**. Deploying application services in a network requires deciding the location of the compute and storage resources needed to run the service. For instance, in edge computing, an extended reality (XR) server could be deployed at the distributed unit (DU), the central unit (CU), the mobile core (MC) or the central cloud [PETERSON]. Bottleneck structures can be used to measure the effect of placing a service on each of the candidate locations, helping the application service provider to make optimized decisions.

* **Multi-job scheduling**. Running a job on a network implies a number of flows will be initiated and terminated throughout the execution of the job. The ripple effects generated from the execution of a job can also be measured using bottleneck structures. This can be used to decide when to optimally launch one or more jobs. For instance, in a data center, bottleneck structure analysis can help the application decide how to optimally schedule multiple AI training or inference jobs that are sharing the same interconnect [G2-SIGCOMM].

* **Link capacity upgrades**. In capacity planning, operators often have a fixed budget and need to decide how to optimally add capacity to a network in order to maximize its performance. The effect of a link upgrade operation can be computed as a derivative with respect to a change (an increase) in the capacity of a link. Through the processing of historical flow information from the network (e.g., NetFlow logs), bottleneck structures can efficiently compute the effect of each link upgrade and identify those that yield maximal performance.

* **Path shortcuts**. Operators in wide area networks need to decide whether a communication path should be set up as purely optical (bypassing layer 3 routing) or undergo an optical-to-electrical-to-optical (OEO) conversion at certain routers in order to perform layer 3 routing [SH-SIGCOMM]. The trade-off is one of cost-efficiency versus better routing control of the network. Bottleneck structures can be used to search for paths that are optimally suitable for being offloaded to a purely optical path. These are also known in the literature as path shortcuts [SH-SIGCOMM].
4. ALTO Bottleneck Structure Service Use Cases

Applications of bottleneck structure analysis expand through a broad class of optimization problems that include traffic engineering, routing, flow scheduling, resiliency analysis, network slicing, service level agreement (SLA) management, network design and capacity planning, to name only a few. In this section, we briefly describe some of the use cases that relate to the objectives of the IETF ALTO Standard.

4.1. Application Rate Limiting for CDN and Edge Cloud Applications

In applications such as CDN, XR or gaming, it is important to throttle the transmission rate of flows to match the true available capacity along their communication path. Transmitting at a lower rate than the available bandwidth leads to lower quality of experience (QoE). Transmitting at a higher rate increases packet losses, which wastes network resources and also leads to a lower QoE.

Estimating the available bandwidth for a flow is complex because it depends on multiple factors including the network topology, the routing configuration and the set of dynamic flows using the network resources. Bottleneck structures capture in a single digraph these three factors, creating a model that allows to estimate the performance of each flow. See for instance Sections 3.1 and 3.2 in [G2-TREP] for examples on how bottleneck structures can be used to estimate the available bandwidth of an application.

An ALTO server could help the application service provider obtain the available bandwidth on a given path by exposing the bottleneck structure of the network. With this information alone, the provider could directly obtain the available bandwidth. Alternatively, the application service could query the ALTO server by passing the path for which the available bandwidth needs to be computed, and the ALTO server could return this value without the need to share the complete bottleneck structure.

4.2. Time-bound Constrained Flow Acceleration for Large Data Set Transfers

Bulk data transfer is an important application to both commercial and government supported networks. For instance, Google’s B4 network supports large-scale data push synchronizing state across multiple global data centers [B4-SIGCOMM]. Another common use case is found in science networks, where massive data sets such as those originated from the Large Hadron Collider at CERN, Switzerland, need to be shared with scientific labs around the world. In this section, we show how bottleneck structures can be used to reconfigure a network.
towards accelerating a given data transfer with the goal to meet a certain time constraint.

To illustrate this use case, we will assume the simple bottleneck structure shown in Figure 3.

Figure 3: Reducing the rate of flow f1 maximally accelerates flow f5.
Suppose our goal is to accelerate flow f5. To achieve this objective, we will also assume that we are allowed to traffic shape (reduce) the rate of any of the other flows. Effectively, for each flow fi different than f5, we need to compute the following derivative

\[-dr_5/d_r_i\]

and then pick the maximum value. Note that in the above expression, we take the left-derivative (d_), since a traffic shaper reduces the rate of a flow. We also negate the derivative (-d), since we are interested in a positive impact induced on flow f5 when reducing the rate of another flow fi.

Such a calculation can be efficiently performed using the bottleneck structure. As an example, Figure 3 illustrates how the value of (-dr5/d_r1) is computed. First, we reduce the rate of flow f1 by 1 unit. This perturbation propagates through the bottleneck structure reaching flow f5 via two paths:

11 -> f2 -> 12 -> f3 -> 14 -> f5
11 -> f2 -> 13 -> f4 -> 14 -> f5

Using the link and flow equations (Section 3.6), each path simply flips the sign of the perturbation every time a link vertex is traversed. (The reason why the sign is flipped at each link vertex is explained by the link and flow equations that dictate how perturbations propagate through the bottleneck structure. Further mathematical descriptions to explain this effect are outside the scope of this document. For detailed mathematical derivations and additional examples, please see [G2-TREF]).

When reaching vertex f5, we find that each path contributes 1 unit of bandwidth. Thus we have:

\[-dr_5/d_r_1 = 1 + 1 = 2\]

In fact, it can be seen that this derivative is maximal. That is, traffic shaping any other flow would yield a smaller increase in the rate of f5. Thus, an operator can conclude that traffic shaping flow f1 yields an optimal strategy to maximally accelerate the rate of flow f5. Note also that in this case, there is a positive multiplier effect, since reducing flow f1’s rate by 1 unit, leads to an increase on flow f5’s rate by more than 1 unit. This is known as a power gradient [G2-SIGCOMM].
While left outside the scope of this document, bottleneck structures can also be used to efficiently compute the value of the optimal traffic shaper (i.e., in our example, to find by how much we should traffic shape flow f1) and to quantify the impact on the flow being accelerated. This information can also be used by the application to estimate the flow's completion time.

An ALTO server could help the application service provider identify an optimized traffic shaping strategy by exposing the bottleneck structure of the network. With this information alone, the provider could efficiently compute an optimized set of traffic shapers. Alternatively, the application service could query the ALTO server by passing the set of flows that are allowed to be traffic shaped and the flow that needs to be accelerated, and the ALTO server could return the set of recommended traffic shapers.

4.3. Application Performance Optimization Through AI Modeling

A relevant and emerging area in the field of application performance is AI-based network modeling. Several global initiatives are been undertaken to apply AI to the field of understanding and predicting network performance. For instance, OpenNetLab [BE-ONL] provides a distributed networking platform with many collaborative nodes (universities and companies) and common benchmarking datasets for researchers to collect real networking data and train their AI models for various networking environments, including the Internet, cloud, and wireless networks. There also exist global benchmarks and challenges to foster innovation in this field, such as the ACM MMSys Challenge [MMSYS], which focuses on novel AI-based bandwidth estimation algorithms to enable superior overall QoE on a global production testbed built for real-time communications (RTC) of video and audio.

Modeling communication networks using purely AI frameworks such as deep learning is challenging as it requires very large production data sets that often times are not available. They also require many years of intense, global collaborative R&D. To address these challenges, it has been seen that hybrid models built by combining AI with a "physics" model of the network can outperform purely AI models, as they may require less training data to achieve the same or better performance. For instance, the top two winners of the ACM MMSys 2021 Bandwidth Prediction Challenge [MMSYS] were based on hybrid models.

Because bottleneck structures provide a "physics" model of the network that can both qualify and quantify the forces of interactions among flows and links, they can be used in combination with AI to enable better performance than purely AI-based models. For instance,
this area is being discussed in the IETF ALTO WG (e.g., [BE-ONL]) as a potential use case in the ALTO Standard to help optimize the performance of RTC applications. In particular, a key building block to optimize the QoE performance of RTC applications is the bandwidth estimation module. This module runs on the endpoint of a real-time video application and aims at dynamically adapting the video bitrate to stay within the available network capacity. A limitation in the current algorithms, however, is their lack of network state visibility. This requires the algorithms to rely entirely on local indicators such as packet loss or latency, which leads to poor training and inference performance. Information provided by the bottleneck structure (which includes topological, routing and flow information of the network in a single digraph) exposed via the ALTO service could help unlock a richer set of machine learning algorithms to optimize application performance.

An ALTO server could help the application service provider implement AI-assisted prediction algorithms by exposing the bottleneck structure of the network. Alternatively, ALTO could implement an AI-assisted prediction module with the help of bottleneck structures. The application would then query the ALTO server to obtain the predicted value.

4.4. Optimized Joint Routing and Congestion Control

In traditional IP networks, the problems of flow routing and congestion control are separately resolved by following a two-step process: first, a routing protocol is used to determine the path between any two nodes in a network; then, flows are routed according to such paths and their transmission rates are regulated using a congestion control algorithm. This layered and disjoint approach is known to be scalable but suboptimal because the routing algorithm identifies paths without taking into account the flow transmission rates assigned by the congestion control algorithm.

Suppose that an application is trying to launch a new flow between two endpoints with the goal to maximize the available bandwidth. One can be tempted to think that, to identify the path with maximal available bandwidth, it suffices to look at the current state of the network and find the least congested path offering the highest capacity. This approach, however, is naive since it does not take into account the fact that the placement of the new flow onto the network will itself create a perturbation in the network, potentially making the chosen path suboptimal or, even more troublesome, negatively affecting the performance of other priority flows.
The goal of the joint routing and congestion control problem between two given endpoints E1 and E2 consists in finding the path from E1 to E2 that will yield the highest throughput _after_ the flow is placed on the network (i.e., taking into account the effect of placing the flow).

The solution to this problem is introduced in [G2-TREP] by employing a strategy that combines the strengths of both the Dijkstra algorithm and the insights revealed by the bottleneck structure. The algorithm can both compute the optimal path and measure the overall network-wide impact of deploying the new flow on the path. It also enables a framework to identify new good-performing paths that have a limited negative impact on the rest of the flows in the network. This allows network and application providers to identify paths that can both provide good performance to the newly added application flow while preserving the performance of the existing high-priority flows.

An ALTO server could help the application service provider optimize the path selection decision by exposing the bottleneck structure of the network. With this information alone, the provider could efficiently compute the optimal path (e.g., using the algorithm introduced in [G2-TREP]). Alternatively, the application service could query the ALTO server by passing the information of the two endpoints that need to be connected, and the ALTO server could return a list of the top-N paths with the highest throughput and their expected performance.

4.5. Service Placement for Edge Computing

Determining the proper location to deploy an application service in the edge cloud is critical to ensure a good quality of experience (QoE) for its users. Yet the service placement problem is known to be NP-Hard [JSP-INFOCOM], requiring heuristics to compute good (albeit suboptimal) solutions.

In [G2-SIGCOMM], it is shown that Bottleneck structures can also be used as highly scalable network simulators to evaluate the performance of a network reconfiguration such as the placement of a new service on a edge cloud. In particular, bottleneck structures can very efficiently (1) compute the performance of each flow in the network and (2) quantify the effects of the arrival (departure) of new (existing) flows to (from) the network. This allows to simulate the full transmission of an application traffic pattern very efficiently, three or more orders of magnitude faster than traditional packet simulators.

Network and application providers can use this capability in two ways:
* Given a set of possible placement strategies, bottleneck structures can be used to simulate them in real time, helping the operator select the one that provides the best performance while guaranteeing the service level agreements (SLAs) of the other existing applications.

* Despite the server placement problem being intractable, bottleneck structures provide a framework to identify good candidate solutions. In particular, by capturing the topology, routing, and flow information in a single computational graph, they can be used to efficiently explore directions in the feasible set that yield incrementally better performance. By moving in these incremental directions, the placement algorithm can progress within the enormous feasible set towards the optimal solution.

An ALTO server could help the application service provider optimize the placement decision by exposing the bottleneck structure of the network. With this information alone, the provider could compute the effect of placing the service in one location versus another. Alternatively, the application service could query the ALTO server by passing the information of the possible locations where it can be placed, and the ALTO server could return an ordered list of the locations and their expected performance.

4.6. Training Neural Networks and AI Inference for Edge Clouds, Data Centers and Planet-Scale Networks

Neural network training and inference using distributed computing systems are the subject of intense research and one of the leading target applications in today’s communication networks. [TOPOOPT-MIT] [FLEXFLOW-STFORD] [SINGULARITY-MSFT]. To illustrate this use case, we will focus our discussion on three types of networks: edge clouds, data centers and planet-scale networks.

5G and Edge Clouds enable for the first time the ability to provide intelligence at the edge of the network. This capability is disruptive in that humans and machines will have access to unprecedented compute power to perform AI inference in real time. For instance, using augmented reality (AR), humans will be able to make better informed decisions as they navigate through an environment by leveraging AI-inference on video and audio signals captured in real time from their user equipments (UEs). Similarly, machines such as vehicles or factory robots will be able to use AI inference to optimize their actions.

Two resources are needed to perform inference: (1) Input data from the environment (e.g., image and audio signals captured from a video camera) and (2) compute (typically in the form of GPUs and CPUs).
The input data needs to be transmitted from the location where it is captured (e.g., a micro-camera running on a human’s glasses) to the location where it is to be processed for inference. The transmission of the input data requires communication resources, whereas the inference process requires computing resources. Since computing resources in the edge cloud (Figure 4) are distributed across the user equipment (UE), the radio unit (RU), the distributed unit (DU) and the central unit (CU) [PETERSON], the problem of efficiently performing AI-inference is one of optimizing the trade-off between communication and compute as follows: compute (communication) power is more scarce (abundant) if the inference is performed closer to the UE, and more abundant (scarce) if performed closer to the CU. For instance, if an AR application running on a UE needs to perform an inference task at a time when the communication path from the RU to the DU is highly congested, then it will have an incentive to perform such a task directly in the UE or in the RU. If instead the network offers an uncongested path to the DU and the CU, it will have an incentive to run the inference task on these other nodes since they offer more compute power.

![Diagram of AI-inference application in the edge cloud](image)

Figure 4: An AI-inference application in the edge cloud needs to place the inference task on a compute node location (UE, RU, DU or CU) that will perform well from both a compute and a communication standpoint.

Using ALTO path vector [I-D.ietf-alto-path-vector] and performance metrics [I-D.ietf-alto-performance-metrics] features, the application could retrieve the amount of compute resources located in the RU, DU and CU. By extending ALTO to support bottleneck structures, the application would also be able to estimate in real-time the available bandwidth for the paths UE-RU, UE-RU-DU, and UE-RU-DU-CU. Further, using bottleneck structure methods described in [G2-SIGCOMM], the application would be able to estimate the time to complete the inference task for each of the four possible scenarios (running in the UE, the RU, the DU or, or the CU) and choose the configuration with the fastest execution.

Similar joint compute-communication optimization problems appear when performing neural network training in large-scale data centers. Large-scale data centers with millions of compute nodes are used to train gigantic neural networks (with potentially trillions of
parameters). Such a massive task needs to be broken down into smaller subtasks that are then executed on the nodes. Once again, compute and communication need to be jointly optimized (see [TOPOOPT-MIT] and [FLEXFLOW-STFORD]) in order to ensure regions in the network don’t become bottlenecks. By exposing bottleneck structure information using ALTO, the AI-training application can make better subtask placement decisions that avoid potential network bottlenecks.

Finally, AI-training using planet-scale networks generalizes the same joint compute and communication problem to an Internet level [SINGULARITY-MSFT], with the need to implement a global scheduler that is responsible for placing workloads onto clusters of globally-distributed compute nodes. Here too enabling better network state visibility using ALTO and bottleneck structure graphs could help the scheduler make better task placement decisions.

5. Example: Application Layer Traffic Optimization using Bottleneck Structures

In this section we provide an example illustrating how bottleneck structures can be used to optimize application performance. This example will then be referenced in Section 6 to discuss and introduce the necessary requirements to integrate the BSG service into the ALTO standard. It is worth noticing that, as shown in Section 4, bottleneck structures have numerous applications. This section provides a complete example for just one of the use cases. In particular, the focus of the next example is on the joint routing and congestion control use case Section 4.4.

Figure 5 provides a view of Google’s B4 network as presented in [B4-SIGCOMM], providing connectivity to 12 data centers distributed across the world (two in Asia, six in America and four in Europe).
Figure 5: Google’s B4 network introduced in [B4-SIGCOMM].

The 12 data centers are connected via a total of 19 links, labeled l1, l2, ... l19. Table 1 presents the pair of data centers that each link is connected to.
<table>
<thead>
<tr>
<th>Link</th>
<th>Adjacent data centers</th>
<th>Link</th>
<th>Adjacent data centers</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>DC1, DC2</td>
<td>111</td>
<td>DC10, DC12</td>
</tr>
<tr>
<td>12</td>
<td>DC1, DC3</td>
<td>112</td>
<td>DC4, DC5</td>
</tr>
<tr>
<td>13</td>
<td>DC3, DC4</td>
<td>113</td>
<td>DC5, DC6</td>
</tr>
<tr>
<td>14</td>
<td>DC2, DC5</td>
<td>114</td>
<td>DC11, DC12</td>
</tr>
<tr>
<td>15</td>
<td>DC3, DC5</td>
<td>115</td>
<td>DC4, DC7</td>
</tr>
<tr>
<td>16</td>
<td>DC6, DC7</td>
<td>116</td>
<td>DC4, DC8</td>
</tr>
<tr>
<td>17</td>
<td>DC7, DC8</td>
<td>117</td>
<td>DC7, DC8</td>
</tr>
<tr>
<td>18</td>
<td>DC8, DC10</td>
<td>118</td>
<td>DC9, DC11</td>
</tr>
<tr>
<td>19</td>
<td>DC9, DC10</td>
<td>119</td>
<td>DC10, DC11</td>
</tr>
<tr>
<td>110</td>
<td>DC7, DC11</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Table 1: Link connectivity (adjacency matrix) in the B4 network.

For the sake of illustration, we will assume a simple configuration consisting of a pair of flows (one for each direction) connecting every data center in the US with every data center in Europe, with all flows routed along a shortest path from source to destination. Since there are six data centers in the US and four in Europe, this configuration has a total of 48 flows. All links are assumed to have a capacity of 10 Gbps except for the transatlantic links (DC7-DC11 and DC8-DC10), which are configured at 25 Gbps.

The next Figure presents the bottleneck structure of Google’s B4 network with the assumed flow configuration. Please see Section 3.1 for a description of how to interpret the bottleneck structure. (See also [G2-SC], [G2-TREP] for details on the algorithm used to compute the bottleneck structure.)
For the sake of compactness, Figure 6 only includes the bottleneck links and a subset of the flow vertices that are part of the complete bottleneck structure. In particular, out of the 19 links that are part of B4, six links (l15, l7, l8, l10, l5, l3) are bottlenecks.

The bottleneck structure graph shows the existence of two bottleneck levels in our configuration example:
The first level at the top of the bottleneck structure includes links l15 and l7. Flows f1, f2, f3, f10, etc. are bottlenecked at this level.

The second level of the bottleneck structure includes links l8, l10, l5 and l3. Flows f6, f9, f23, f18, etc. are bottlenecked at this level.

From the MBA Property (Property 3), we know that flows bottlenecked by a link at level 2 will enjoy higher available bandwidth than flows bottlenecked at level 2. For instance, consider the following directed path in the bottleneck structure:

\[
115 \rightarrow f1 \rightarrow 18 \rightarrow f6
\]

Using the MBA property, we have that since 115 precedes 18, it must be that \( s_{15} < s_{8} \), where \( s_{15} \) is the rate of flow f1 bottlenecked at 115 and \( s_{8} \) is the rate of flow f6 bottlenecked at 18.

Suppose now that an application needs to place a new flow on Google’s B4 network to transfer a large data set from data center 11 (DC11) to data center 4 (DC4). The application needs to select and configure a path from DC11 to DC4 (for instance, this could be done by using SR [RFC8402]). The shortest path DC11 -> 110 -> DC7 -> 115 -> DC4 is often used as the default option. Doing so, however, implies that the flow will be bottlenecked at link 115 at the upper level of the bottleneck structure, leading to a lower transmission rate. If instead we choose the non-shortest path DC11 -> 19 -> DC10 -> 18 -> DC8 -> 116 -> DC4, now the flow will be bottlenecked at link 18 (at the lower level of the bottleneck structure), leading to a higher transmission rate.

Using QTBS, we can also numerically compute the transmission rate of the flow on each of the two path options. (See Section 3.1 in [G2-TREP] for a detailed description of how to compute the transmission rate assigned to the flow on each of these paths.) In particular, we obtain that when the application chooses the shortest path (bottlenecked at level 1 of the bottleneck structure), it gets a transmission rate of 1.429 Gbps. If instead the application chooses the slightly longer path (bottlenecked at level 2 of the bottleneck structure), then it gets a transmission rate of 2.5 Gbps, an increase of 74.95% with respect to the shortest path solution.

[G2-TREP] introduces also a very efficient routing algorithm that uses the bottleneck structure to find the maximal throughput path for a flow in \( O(V+E \times \log(V)) \) steps, where V is the number of routers and E is the number of links in the network.
Overall, this example illustrates that, equipped with knowledge about the bottleneck structure of a network, an application can make better informed decisions on how to route a flow. In the next sections, we will use this example to support a discussion on the requirements for integrating the Bottleneck Structure Graph (BSG) service into the ALTO standard.

6. Requirements

This section provides a discussion on the necessary requirements to integrate the BSG service into the ALTO standard.

6.1. Requirement 1: Bottleneck Structure Graph (BSG) Abstraction

The first base requirement consists in extending the ALTO server with the capability to compute bottleneck structures. For instance, with this capability, given the network configuration in Figure 5, the ALTO server would be able to compute the bottleneck structure shown in Figure 6:

* Requirement 1A (R1A). The ALTO server MUST compute the bottleneck structure graph to allow applications optimize their performance using the BSG service.

We note that the alternative, which would consists in ALTO simply providing the necessary information for applications to compute their own bottleneck structures, would not scale due to the following issues:

* Suppose that 1 ALTO server is providing support to N ALTO clients. Then, requiring each application to compute the bottleneck structure would imply performing N identical and redundant computations. By computing the bottleneck structure in the ALTO server, a one-time computation can be leveraged by all N clients. We also note that [G2-SC] demonstrates that bottleneck structures can be efficiently computed in real time by the server even for large scale networks.

* A production-ready high-speed implementation of QTBS is relatively sophisticated. Requiring the applications to implement their own QTBS optimization library would impose unnecessary and (perhaps more importantly) out-of-scope requirements to the application, which should focus on providing its service rather than implementing a network optimization math library.

The next requirement focuses on the type of bottleneck structure an ALTO server must compute:
* Requirement 1B (R1B). The ALTO server MUST at least support the computation of one bottleneck structure type from Section 3.7. Depending on the network information available (e.g., presence of QoS class information), the ALTO server MAY support all the three bottleneck structure types, in which case the ALTO client MAY be able to choose the bottleneck structure type for retrieval. Also, it is RECOMMENDED that the ALTO server supports the computation of the path gradient graph (PGG) as the default bottleneck structure implementation for retrieval by the ALTO clients.

6.2. Requirement 2: Information Received from the Network

To compute a bottleneck structure, two pieces of information are required:

* Topology Object (T). The T Object is a data structure that includes:
  
  (1) A Topology Graph (V, E), where V is the set of routers and E is the set of links connecting the routers in the network.

  (2) A Capacity Dictionary (C), a key-value table mapping each link with its capacity (in bps).

* Flow Dictionary (F). The F Dictionary is a key-value table mapping every flow with the set of links it traverses.

As shown in [G2-TREP], the above information is enough to compute the bottleneck structure. In fact, with only the F and C dictionaries, one can compute the bottleneck structure. The topology graph (V, E) is needed to perform optimal routing computations (for instance, to find new paths in the network that yield higher throughput, as illustrated in Section 5).

The above discussion leads to the following requirement:

* Requirement 2A (R2A). The ALTO server MUST collect information about (1) the set of routers and links in a network, (2) the capacity of each link and (3) the set of links traversed by each flow.
Information about the set of routers, links and link capacity is typically available from protocols and technologies such as SNMP, BGP-LS, SDN, or domain specific topology logs. This information is enough to construct the T Object. Information about the set of links traversed by each flow can be obtained from protocols such as NetFlow, sFlow, IPFIX, etc. See [FLOWDIR] and [G2-SC] for examples of how requirement R2A is implemented in real-world production networks.

6.3. Requirement 3: Information Passed to the Application

The following requirement is necessary so that applications can optimize their performance using bottleneck structure information:

* Requirement 3A (R3A). The ALTO client MUST be able to query the ALTO server to obtain the current bottleneck structure of the network, represented as a digraph.

In addition, the current ALTO services can be extended with additional information obtained from the bottleneck structure:

* Requirement 3B (R3C). One or more ALTO services (the Network Map, the Cost Map, the Entity Property Map or the Endpoint Cost Map) MUST support reporting to ALTO clients additional network state information derived from the bottleneck structure to the ALTO client.

For example, the ALTO Cost Map Service can be extended with a new cost metric that corresponds to the estimated available bandwidth between two endpoints according to the bottleneck structure model.

6.4. Requirement 4: Features Needed to Support the Use Cases

Bottleneck structures offer a rich framework to optimize application performance for a variety of use cases. In addition to the base requirement to construct the bottleneck structure graph (see R1A), in this section we enumerate additional capabilities that must be supported by the ALTO BSG service to ensure it is effective in helping applications optimize their performance for each of the supported use cases.

* Requirement 4A (R4A). The ALTO BSG service MUST be able to compute the effect of network reconfigurations using bottleneck structure analysis and according to the types described in Section 3.9.
For example, an extended reality (XR) application might need to choose where to place a containerized instance of the XR service among a set of possible server racks located in various edge cloud locations. The application would query the ALTO BSG service to obtain the projected performance results of placing the new service instance on each possible location, allowing it to select the one that would yield the highest performance.

The following requirement is necessary to ensure that the information provided by the BSG service is not stale:

Requirement 4B (R4B). The BSG service MUST be able to update the bottleneck structure graph in near-real time, at least once a minute or less.

In [G2-SC] it is shown that bottleneck structures can be computed in a fraction of a session for a production US wide network with about 100 routers and 500 links. Thus, the above requirement should be achievable with a good implementation of the bottleneck structure algorithm [G2-TREP].

7. Security Considerations

Future versions of this document may extend the base ALTO protocol, so the Security Considerations [RFC7285] of the base ALTO protocol fully apply when this proposed extension is provided by an ALTO server.

The Bottleneck Structure Graph extension requires additional scrutiny on three security considerations discussed in the base protocol: Confidentiality of ALTO information (Section 15.3 of [RFC7285]), potential undesirable guidance from authenticated ALTO information (Section 15.2 of [RFC7285]), and availability of ALTO service (Section 15.5 of [RFC7285]).

For confidentiality of ALTO information, a network operator should be aware that this extension may introduce a new risk: As the Bottleneck Structure information may reveal more fine-grained internal network structures than the base protocol, an attacker may identify the bottleneck link and start a distributed denial-of-service (DDoS) attack involving minimal flows to conduct in-network congestion. Given the potential risk of leaking sensitive information, the BSG extension is mainly applicable in scenarios where:

* The properties of the Bottleneck Structure Graph do not impose security risks to the ALTO service provider, e.g., by not carrying sensitive information.
The ALTO server and client have established a reliable trust relationship, for example, operated in the same administrative domain, or managed by business partners with legal contracts and proper authentication and privacy protocols.

The ALTO server implements protection mechanisms to reduce information exposure or obfuscate the real information. Implementations involving reduction or obfuscation of the Bottleneck Structure information SHOULD consider reduction/obfuscation mechanisms that can preserve the integrity of ALTO information, for example, by using minimal feasible region compression algorithms [NOVA] or obfuscation protocols RESA [MERCATOR]. We note that these obfuscation methods are experimental and their practical applicability to the generic capability provided by this extension is not fully assessed.

We note that for operators that are sensitive about disclosing flow-level information (even if it is anonymized), then they SHOULD consider using the Path Gradient Graph (PGG) or the QoS-Path Gradient Graph (Q-PGG) since these objects provide the additional security advantage of hiding flow-level information from the graph.

For undesirable guidance, the ALTO server must be aware that, if information reduction/obfuscation methods are implemented, they may lead to potential misleading information from Authenticated ALTO Information. In such cases, the Protection Strategies described in Section 15.2.2 of [RFC7285] MUST be considered.

For availability of ALTO service, an ALTO server should be cognizant that using Bottleneck Structures might have a new risk: frequently querying the BSG service might consume intolerable amounts of computation and storage on the server side. For example, if an ALTO server implementation dynamically computes the Bottleneck Structure for each request, the BSG service may become an entry point for denial-of-service attacks on the availability of an ALTO server.

To mitigate this risk, an ALTO server may consider using optimizations such as precomputation-and-projection mechanisms [MERCATOR] to reduce the overhead for processing each query. An ALTO server may also protect itself from malicious clients by monitoring the behaviors of clients and stopping serving clients with suspicious behaviors (e.g., sending requests at a high frequency).

8. IANA Considerations

Future versions of this document may register new entries to the ALTO Cost Metric Registry, the ALTO Cost Mode Registry, the ALTO Domain Entity Type Registry and the ALTO Entity Property Type Registry.
9. References

9.1. Normative References

[I-D.ietf-alto-path-vector]

[I-D.ietf-alto-performance-metrics]


9.2. Informative References


Bandwidth Estimation on OpenNetLab", IETF Plenary 112, IETF ALTO WG, 2021,


Poularakis et al., D., "Joint Service Placement and Request Routing in Multi-cell Mobile Edge Computing Networks", n.d..

Lorenz, E., "Does the flap of a butterfly's wings in Brazil set off a tornado in Texas?", American Association for the Advancement of Science, 139th Meeting, 1972.


Authors’ Addresses

Jordi Ros-Giralt
Qualcomm
Email: jros@qti.qualcomm.com
A Cost Mode Registry for the Application-Layer Traffic Optimization (ALTO) Protocol
draft-ietf-alto-cost-mode-05

Abstract

This document creates a new IANA registry for tracking cost modes supported by the Application-Layer Traffic Optimization (ALTO) Protocol. Also, this document relaxes a constraint that was imposed by the ALTO specification on allowed cost mode values.

This document updates RFC 7285.

Editorial Note (To be removed by RFC Editor)

Please update RFC XXXX statements within the document with the RFC number to be assigned to this document.

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 https://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 4 December 2022.

Copyright Notice

Copyright (c) 2022 IETF Trust and the persons identified as the document authors. All rights reserved.
1. Introduction

The cost mode attribute indicates how costs should be interpreted when communicated in the Application-Layer Traffic Optimization (ALTO) Protocol [RFC7285]. The base ALTO specification includes a provision for only two modes:

"numerical": Indicates that numerical operations can be performed (e.g., normalization) on the returned costs (Section 6.1.2.1 of [RFC7285]).

"ordinal": Indicates that the cost values in a cost map represent ranking (relative to all other values in a cost map), not actual costs (Section 6.1.2.2 of [RFC7285]).

Additional cost modes are required for specific ALTO deployment cases (e.g., [I-D.ietf-alto-path-vector]). In order to allow for such use cases, this document relaxes the constraint imposed by the base ALTO specification on allowed cost modes (Section 3) and creates a new ALTO registry to track new cost modes (Section 5).
The mechanisms defined in [RFC7285] are used to advertise the support of new cost modes for specific cost metrics. Refer to Section 4 for more details.

2. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

This document makes use of the terms defined in [RFC7285].

3. Updates to RFC7285

3.1. Updates to Section 6.1.2 of RFC7285

This document updates Section 6.1.2 of [RFC7285] as follows:

OLD:

The cost mode attribute indicates how costs should be interpreted. Specifically, the cost mode attribute indicates whether returned costs should be interpreted as numerical values or ordinal rankings.

It is important to communicate such information to ALTO clients, as certain operations may not be valid on certain costs returned by an ALTO server. For example, it is possible for an ALTO server to return a set of IP addresses with costs indicating a ranking of the IP addresses. Arithmetic operations that would make sense for numerical values, do not make sense for ordinal rankings. ALTO clients may handle such costs differently.

Cost modes are indicated in protocol messages as strings.

NEW:

The cost mode attribute indicates how costs should be interpreted. Two cost modes (numerical values and ordinal rankings) are defined, but additional cost modes can be defined in the future.

It is important to communicate such information to ALTO clients, as certain operations may not be valid on certain costs returned by an ALTO server. For example, it is possible for an ALTO server to return a set of IP addresses with costs indicating a ranking of
the IP addresses. Arithmetic operations that would make sense for numerical values, do not make sense for ordinal rankings. ALTO clients may handle such costs differently.

Cost modes are indicated in protocol messages as strings.

Future documents that define a new cost mode are strongly recommended to indicate whether that new cost mode applies to all or a subset of cost metrics. This recommendation is meant to prevent non-deterministic behaviors that may result in presenting a cost mode with a specific metric, while such an association does not make sense or can’t be unambiguously interpreted by ALTO implementations.

If the definition of a cost mode does not indicate whether that cost mode applies to a subset of cost metrics, ALTO implementations MUST be prepared to accept that cost mode for any cost metric.

3.2. Updates to Section 10.5 of RFC7285

This document updates Section 10.5 of [RFC7285] as follows:

OLD:

A cost mode is encoded as a string. The string MUST have a value of either "numerical" or "ordinal".

NEW:

A cost mode is encoded as a string. The string MUST be no more than 32 characters, and it MUST NOT contain characters other than US-ASCII alphanumeric characters (U+0030-U+0039, U+0041-U+005A, and U+0061-U+007A), the hyphen-minus ('-', U+002D), the colon (':', U+003A), or the low line ('_', U+005F). Cost modes reserved for Private Use are prefixed with "priv:" (Section 5). Otherwise, the cost mode MUST have a value that is listed in the registry created in Section 5 of RFCXXXX.
4. Backward Compatibility Considerations

ALTO servers that support new cost modes for specific cost metrics will use the mechanism specified in Section 9.2 of [RFC7285] to advertise their capabilities. ALTO clients (including legacy) will use that information to specify cost constraints in their requests (e.g., indicate a cost metric and a cost mode). An example of such behavior is depicted in Section 9.2.3 of [RFC7285].

If an ALTO client includes a cost mode that is not supported by an ALTO server, the server indicates such an error with the error code E_INVALID_FIELD_VALUE as per Section 8.5.2 of [RFC7285]. In practice, legacy ALTO servers will reply with the error code E_INVALID_FIELD_VALUE to requests that include a cost type other than "numerical" or "ordinal" for the "routingcost" cost metric.

The encoding constraints in Section 3.2 do not introduce any interoperability issue given that currently implemented cost modes adhere to these constrains (mainly, those in [RFC7285] and [I-D.ietf-alto-path-vector]).

5. IANA Considerations

This document requests IANA to create a new subregistry entitled "ALTO Cost Modes" under the "Application-Layer Traffic Optimization (ALTO) Protocol" registry available at [ALTO].

The assignment policy for this subregistry is "IETF Review" (Section 4.8 of [RFC8126]).

Requests to register a new ALTO cost mode must include the following information:

Identifier: The name of the ALTO cost mode. Refer to Section 3.2 for more details on allowed encoding.

Description: A short description of the requested ALTO cost mode.

Intended Semantics: A reference to where the semantic of the requested cost mode is defined.

Reference: A reference to the document that registers the requested cost mode.

Cost modes prefixed with "priv:" are reserved for Private Use (Section 4.1 of [RFC8126]). This document requests IANA to add the following note to the new subregistry:
Note: Identifiers prefixed with ‘priv:’ are reserved for Private Use (see [RFCXXXX], Section 5).

The subregistry is initially populated with the following values:

<table>
<thead>
<tr>
<th>Identifier</th>
<th>Description</th>
<th>Intended Semantics</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>numerical</td>
<td>Indicates that numerical operations can be performed on the returned costs</td>
<td>Section 6.1.2.1 of RFC7285</td>
<td>RFCXXXX</td>
</tr>
<tr>
<td>ordinal</td>
<td>Indicates that the cost values in a cost map represent ranking</td>
<td>Section 6.1.2.2 of RFC7285</td>
<td>RFCXXXX</td>
</tr>
</tbody>
</table>

6. Security Considerations

This document does not introduce new concerns other than those already discussed in Section 15 of [RFC7285].

7. Acknowledgements

Many thanks to Benjamin Kaduk for spotting the issue during the review of [I-D.ietf-alto-path-vector].

Thanks to Adrian Farrel, Dhruv Dhody, Luis Miguel Contreras Murillo, Sabine Randriamasy, and Qiao Xiang for the review and comments.

Special thanks to Kai Gao for Shepherding the document.

Thanks to Martin Duke for the AD review.

Thanks to Roni Even for the gen-art review, Jaime Jimenez for the artart review, and Stephen Farrell for the secdir review.

Thanks to Robert Wilton, Lars Eggert, Francesca Palombini, Roman Danyliw, Paul Wouters, and Murray Kucherawy for the IESG review.

8. References

8.1. Normative References
8.2. Informative References


[I-D.ietf-alto-path-vector]

Authors’ Addresses

Mohamed Boucadair
Orange
35000 Rennes
France
Email: mohamed.boucadair@orange.com

Qin Wu
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Abstract

The ALTO base protocol [RFC7285] uses HTTP/1.x as the transport protocol and hence ALTO transport includes the limitations of HTTP/1.x. ALTO/SSE [RFC8895] addresses some of the limitations, but is still based on HTTP/1.x. This document introduces ALTO new transport, which provides the transport functions of ALTO/SSE on top of HTTP/2, for more efficient ALTO transport.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

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 September 6, 2022.
1. Introduction

Application-Layer Traffic Optimization (ALTO) provides a means for network applications to obtain network status information. The ALTO base protocol [RFC7285] is based on the sequential request and response model of HTTP/1.1 [RFC7230]; hence, in the base protocol, an ALTO client can issue only a sequence of requests on network information resources, and the ALTO server sends the information resources one-by-one, in the order of the request sequence.

To address the use cases where an ALTO client may need to efficiently monitor changes to a set of network information resources and the protocol is still based on the HTTP/1.1 model, the ALTO Working Group introduces ALTO/SSE (ALTO Incremental Update based on Server-Sent-Event) [RFC8895], so that an ALTO client can manage (i.e., add and remove) a set of requests maintained at an ALTO server, and the
server can continuously, concurrently, and incrementally push updates whenever a monitored network information resource changes. Figure 1 shows the architecture and message flow of ALTO/SSE, which can be considered as a more general transport protocol than the ALTO base transport protocol. Although ALTO/SSE allows the concurrent transport of multiple ALTO information resources, it has complexities and limitations. For example, it requires that the server provide a separate control URI, leading to complexity in management.

---

Figure 1: ALTO SSE Architecture and Message Flow.

This document specifies ALTO new transport, which realizes ALTO/SSE but takes advantage of new HTTP capabilities.

2. ALTO New Transport Design Requirements

The new transport is designed to satisfy a set of requirements. First, it should satisfy the following requirements to realize the functions of ALTO/SSE:

- R0: Client can request any resource using the connection, just as using ALTO base protocol using HTTP/1.x.
- R1: The client can request the addition (start) of incremental updates to a resource.
- R2: The client can request the deletion (stop) of incremental updates to a resource.

Schott, et al. Expires September 6, 2022 [Page 3]
o R3: The server can signal to the client the start or stop of incremental updates to a resource.

o R4: The server can choose the type of each incremental update encoding, as long as the type is indicated to be acceptable by the client.

Following the ALTO framework, the new transport protocol should still be HTTP based:

o R5: The design follows basic principle of HTTP---Representational State Transfer and hence can use only HTTP verbs (GET, POST, PUT, DELETE, HEAD).

o R6: The design takes advantage of HTTP/2 design features such as parallel transfer and respects HTTP/2 semantics such as the semantics of PUSH_PROMISE.

To allow flexible deployment, the new transport protocol should be flexible:

o R7: The design should support capability negotiation.

3. ALTO New Transport Information Structure

A key design of ALTO new transport is to distinguish between information about ALTO resources and information about ALTO transport. It introduces the following transport information structures to distribute ALTO information resources:

o The transport state from the ALTO server to an ALTO client (or a set of clients) for an ALTO information resource is conceptually through a transport queue. A static ALTO information resource (e.g., Cost Map, Network Map) has a single transport queue, and a dynamic ALTO information resource (e.g., Filtered Cost Map) may create a queue for each unique filter request.

o Each transport queue maintains two states: (1) the incremental update message queue TQ-BASE-URI/m, and (2) the recipients set TQ-BASE-URI/r, where TQ-BASE-URI is the base URI pointing to the transport queue, TQ-BASE-URI/m/meta is the list of update messages in the update message queue, and TQ-BASE-URI/m/msg-seq-no is a specific update. A transport queue can be created by a POST URI; a client can delete a transport queue by sending DELETE the TQ-BASE-URI. A client with an interest to receive incremental updates should be in TQ-BASE-URI/r.
3.1. Transport Queue

An ALTO client creates a transport queue using ALTO SSE AddUpdateReq ([RFC 8895] Sec. 6.5). Unless the request has incremental-changes to be false, the client is added to TQ-BASE-URI/r.

```json
object {
    ResourceID   resource-id;
    [JSONString  tag;]
    [Boolean     incremental-changes;]
    [Object      input;]
} AddUpdateReq;
```

Any disconnection between the client and the server will remove the client from the receiver queue; that is, the receiver state is ephemeral. A client can also remove itself by deleting itself from the receiver set.

3.2. Incremental Update Message Queue

When a client joins a transport queue and specifies incremental push updates, the first message pushed from the server to the client is the last independent message in the incremental message queue, unless the client specifies a matching message tag.

3.3. Examples

The first example is client receiving cost map and its updates.
POST /tqs HTTP/2
Host: alto.example.com
Accept: application/alto-transport+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: TBD

{
  "resource-id": "my-routingcost-map"
}

HTTP/2 200 OK
Content-Type: application/alto-transport+json

{"mq": "/updates/streams/2718281828459"}

Note that the example above uses HTTP/1.x notation, and it is straightforward to change to use HTTP/2 notation. We use the short notation for now and will update to the HTTP/2 notation in later revisions.
Client -> server request

HEADERS
- END_STREAM
- END_HEADERS
  :method = POST
  :scheme = https
  :path = /tqs

CONTINUATION
+ END_HEADERS
  host = alto.example.com
  accept = application/alto-transport+json
  content-type = application/alto-updatestreamparams+json
  content-length = TBD

DATA
+ END_STREAM
  {
    "resource-id": "my-routingcost-map"
  }

Server -> client response:

HEADERS
- END_STREAM
+ END_HEADERS
  :status = 200
  content-type = application/alto-transport+json
  content-length = TBD

DATA
+ END_STREAM
  {"mq": "/updates/streams/2718281828459"}

The client can check the status of the transport queue from the same connection:
GET /updates/streams/2718281828459/m/meta HTTP/2
Accept: application/alto-transport+json

HTTP/2 200 OK
Content-Type: application/alto-transport+json

{
  [
    {
      "seq": 101,
      "media-type": "application/alto-costmap+json",
      "tag": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe"
    },
    {
      "seq": 102,
      "media-type": "application/merge-patch+json",
      "tag": "cdf0222x59740b0b2e3f8eb1d4785acd42231bfe"
    },
    {
      "seq": 103,
      "media-type": "application/merge-patch+json",
      "tag": "8eb1d4785acd42231bfecdf0222x59740b0b2e3f",
      "equi-link": "/updates/streams/2718281828459/m/aliase1"
    }
  ]
}

The client can directly request an element in the queue, for example,
GET /updates/streams/2718281828459/m/101 HTTP/2
Accept: application/application/alto-costmap+json

HTTP/2 200 OK
Content-Type: application/alto-costmap+json

{
    "meta": {
        "dependent-vtags": [{
            "resource-id": "my-network-map",
            "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
        }],
        "cost-type": {
            "cost-mode": "numerical",
            "cost-metric": "routingcost"
        },
        "vtag": {
            "resource-id": "my-routingcost-map",
            "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
        }
    },
    "cost-map": {
        "PID1": { "PID1": 1, "PID2": 5, "PID3": 10 },
        "PID2": { "PID1": 5, "PID2": 1, "PID3": 15 },
        "PID3": { "PID1": 20, "PID2": 15 }
    }
}

Note from the transport queue state that the 103 message has an
OPTIONAL link to a complete snapshot, which a client can request.

Instead of directly requesting, the client can wait for the server
for incremental push, where the server first sends PUSH_PROMISE with
the GET URI as above.

A client can leave incremental updates by sending the request:

DELETE /updates/streams/2718281828459/r/self HTTP/2
Accept: application/alto-transport+json

HTTP/2 200 OK

A second client can request the creation for the same resource
and the server can return the same transport queue.
A client can delete the transport queue from its view, and as long as there are other clients, the server will still maintain the transport queue.

```
DELETE /updates/streams/2718281828459 HTTP/2
Accept: application/alto-transport+json

HTTP/2 200 OK
```

The transport queue is not limited to only GET resources. The client can also request a filtered ALTO resource, which is shown in the example below:

```
POST /tqs HTTP/2
Host: alto.example.com
Accept: application/alto-transport+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 382

{
  "resource-id": "my-pv",
  "input": {
    "cost-type": {
      "cost-mode": "array",
      "cost-metric": "ane-path"
    },
    "endpoints": {
      "srcs": [ "ipv4:192.0.2.2" ],
      "dsts": [ "ipv4:192.0.2.89", "ipv4:203.0.113.45" ]
    },
    "ane-properties": [ "maxresbw", "persistent-entities" ]
  }
}

HTTP/2 200 OK
Content-Type: application/alto-transport+json

{"mq": "/updates/streams/2718281828459"}
```

4. ALTO/H2 Information Resource Directory (IRD)

Extending the IRD example in Section 8.1 of [RFC8895], Figure 2 is the IRD of an ALTO server supporting ALTO base protocol, ALTO/SSE, and ALTO/H2.

In particular,
"my-network-map": {
    "uri": "https://alto.example.com/networkmap",
    "media-type": "application/alto-networkmap+json",
},

"my-routingcost-map": {
    "uri": "https://alto.example.com/costmap/routingcost",
    "media-type": "application/alto-costmap+json",
    "uses": ["my-networkmap"],
    "capabilities": {
        "cost-type-names": ["num-routingcost"]
    }
},

"my-hopcount-map": {
    "uri": "https://alto.example.com/costmap/hopcount",
    "media-type": "application/alto-costmap+json",
    "uses": ["my-networkmap"],
    "capabilities": {
        "cost-type-names": ["num-hopcount"]
    }
},

"my-filtered-cost-map": {
    "uri": "https://alto.example.com/costmap/filtered/constraints",
    "media-type": "application/alto-costmap+json",
    "accepts": "application/alto-costmapfilter+json",
    "uses": ["my-networkmap"],
    "capabilities": {
        "cost-type-names": ["num-routingcost", "num-hopcount"],
        "cost-constraints": true
    }
},

"my-simple-filtered-cost-map": {
    "uri": "https://alto.example.com/costmap/filtered/simple",
    "media-type": "application/alto-costmap+json",
    "accepts": "application/alto-costmapfilter+json",
    "uses": ["my-networkmap"],
    "capabilities": {
        "cost-type-names": ["num-routingcost", "num-hopcount"],
        "cost-constraints": false
    }
},

"my-props": {
    "uri": "https://alto.example.com/properties",
    "media-type": "application/alto-endpointprops+json",
    "accepts": "application/alto-endpointpropparams+json",
    "capabilities": {
        "prop-types": ["priv:ietf-bandwidth"]
    }
},

"my-pv": {
  "uri": "https://alto.example.com/endpointcost/pv",
  "media-type": "multipart/related;
    type=application/alto-endpointcost+json",
  "accepts": "application/alto-endpointcostparams+json",
  "capabilities": {
    "cost-type-names": [ "path-vector" ],
    "ane-properties": [ "maxresbw", "persistent-entities" ]
  }
},
"update-my-costs": {
  "uri": "https://alto.example.com/updates/costs",
  "media-type": "text/event-stream",
  "accepts": "application/alto-updatestreamparams+json",
  "uses": ["my-network-map",
    "my-routingcost-map",
    "my-hopcount-map",
    "my-simple-filtered-cost-map"
  ],
  "capabilities": {
    "incremental-change-media-types": {
      "my-network-map": "application/json-patch+json",
      "my-routingcost-map": "application/merge-patch+json",
      "my-hopcount-map": "application/merge-patch+json"
    },
    "support-stream-control": true
  }
},
"update-my-costs-h2": {
  "uri": "https://alto.example.com/updates-h2/costs",
  "media-type": "application/alto-h2",
  "accepts": "application/alto-updatestreamparams+json",
  "uses": ["my-network-map",
    "my-routingcost-map",
    "my-hopcount-map",
    "my-simple-filtered-cost-map"
  ],
  "capabilities": {
    "incremental-change-media-types": {
      "my-network-map": "application/json-patch+json",
      "my-routingcost-map": "application/merge-patch+json",
      "my-hopcount-map": "application/merge-patch+json"
    },
    "support-stream-control": true
  }
}
"update-my-props": {
  "uri": "https://alto.example.com/updates/properties",
  "media-type": "text/event-stream",
  "uses": [ "my-props" ],
  "accepts": "application/alto-updatestreamparams+json",
  "capabilities": {
    "incremental-change-media-types": {
      "my-props": "application/merge-patch+json"
    },
    "support-stream-control": true
  }
},
"update-my-pv": {
  "uri": "https://alto.example.com/updates/pv",
  "media-type": "text/event-stream",
  "uses": [ "my-pv" ],
  "accepts": "application/alto-updatestreamparams+json",
  "capabilities": {
    "incremental-change-media-types": {
      "my-pv": "application/merge-patch+json"
    },
    "support-stream-control": true
  }
}

Note that it is straightforward for an ALTO sever to run HTTP/2 and support concurrent retrieval of multiple resources such as "my-network-map" and "my-routingcost-map" using multiple HTTP/2 streams with the need to introducing ALTO/H2.

The resource "update-my-costs-h2" provides an ALTO/H2 based connection, and this is indicated by the media-type "application/alto-h2". For an ALTO/H2 connection, the client can send in a sequence of control requests using media type application/alto-updatestreamparams+json. The server creates HTTP/2 streams and pushes updates to the client.

5. Security Considerations

The properties defined in this document present no security considerations beyond those in Section 15 of the base ALTO specification [RFC7285].

6. IANA Considerations

IANA will need to register the alto-h2 media type under ALTO registry as defined in [RFC7285].
7. Acknowledgments

The authors of this document would also like to thank many for the reviews and comments.

8. References

8.1. Normative References


8.2. Informative References

Authors’ Addresses

Roland Schott
Deutsche Telekom
Heinrich-Hertz-Strasse 3-7
64295 Darmstadt
Germany

Email: Roland.Schott@telekom.de

Y. Richard Yang
Yale University
51 Prospect St
New Haven, CT 06520
USA

Email: yry@cs.yale.edu

Kai Gao
Sichuan University
Chengdu 201804
China

Email: kgao@scu.edu.cn

Jingxuan Jensen Zhang
Tongji University
4800 Cao’An Hwy
Shanghai 201804
China

Email: jingxuan.n.zhang@gmail.com
Abstract

The ALTO base protocol [RFC7285] uses HTTP/1.x as the transport protocol and hence ALTO transport includes the limitations of HTTP/1.x. ALTO/SSE [RFC8895] addresses some of the limitations, but is still based on HTTP/1.x. This document introduces ALTO new transport, which provides the transport functions of ALTO/SSE on top of HTTP/2, for more efficient ALTO transport.

Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119][RFC8174] when, and only when, they appear in all capitals, as shown here.

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 https://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 24 November 2022.
1. Introduction

Application-Layer Traffic Optimization (ALTO) provides a means for network applications to obtain network status information. The ALTO base protocol [RFC7285] is based on the sequential request and response model of HTTP/1.1 [RFC7230]; hence, in the base protocol, an ALTO client can issue only a sequence of requests on network information resources, and the ALTO server sends the information resources one-by-one, in the order of the request sequence.

To address the use cases where an ALTO client may need to efficiently monitor changes to a set of network information resources and the protocol is still based on the HTTP/1.1 model, the ALTO Working Group introduces ALTO/SSE (ALTO Incremental Update based on Server-Sent-Event) [RFC8895], so that an ALTO client can manage (i.e., add and remove) a set of requests maintained at an ALTO server, and the server can continuously, concurrently, and incrementally push updates whenever a monitored network information resource changes. Figure 1 shows the architecture and message flow of ALTO/SSE, which can be considered as a more general transport protocol than the ALTO base transport protocol. Although ALTO/SSE allows the concurrent transport of multiple ALTO information resources, it has complexities and limitations. For example, it requires that the server provide a separate control URI, leading to complexity in management.

---

Figure 1: ALTO SSE Architecture and Message Flow.
This document specifies ALTO/H2, which realizes ALTO/SSE but takes advantage of new HTTP capabilities provided by HTTP/2 [RFC7540].

2. ALTO/H2 Design Requirements

ALTO/H2 is designed to satisfy a set of requirements. First, it should satisfy the following requirements to realize the functions of ALTO/SSE:

* R0: Client can request any resource using the connection, just as using ALTO base protocol using HTTP/1.x.

* R1: The client can request the addition (start) of incremental updates to a resource.

* R2: The client can request the deletion (stop) of incremental updates to a resource.

* R3: The server can signal to the client the start or stop of incremental updates to a resource.

* R4: The server can choose the type of each incremental update encoding, as long as the type is indicated to be acceptable by the client.

Following the ALTO framework [RFC7285] [RFC7971], ALTO/H2 should still be HTTP based:

* R5: The design follows basic principle of HTTP---Representational State Transfer and hence can use only HTTP verbs (GET, POST, PUT, DELETE, HEAD).

* R6: The design takes advantage of HTTP/2 design features such as parallel transfer and respects HTTP/2 semantics such as the semantics of PUSH_PROMISE.

To allow flexible deployment, the new transport protocol should be flexible:

* R7: The design should support capability negotiation.

3. ALTO/H2 Design Overview

A key design of ALTO new transport is to distinguish between information about ALTO resources and information about ALTO transport. It introduces the following transport information structures to distribute ALTO information resources:
* The transport state from the ALTO server to an ALTO client (or a set of clients) for an ALTO information resource is conceptually through a transport queue. A static ALTO information resource (e.g., Cost Map, Network Map) has a single transport queue, and a dynamic ALTO information resource (e.g., Filtered Cost Map) may create a queue for each unique filter request.

* Each transport queue maintains two states: (1) the incremental update message queue, which includes a sequence of incremental update messages and (2) the receiver set, which includes the set of receivers receiving incremental push updates from the ALTO server.

* The transport queue state is exposed to clients through views; that is, a client can see only a virtual view of the server state.

Figure 2 shows an example illustrating the aforementioned information. Each ALTO client (Client 1, Client 2, Client 3) maintains a single HTTP/2 connection with the ALTO server.
Information Resource:

a) Static resource such as NetworkMap
b) Filterable resource such as FilteredCostMap

Figure 2: ALTO New Transport Information Structure.

tq  = transport queue
tq/uq = incremental updates queue
sq/rs = receiver set
The basic work flow of a client connecting to an ALTO server is the following:

Client opens a connection to the server  
Client opens/identifies a transport queue tq  
Client requests transport queue status of tq  
Client requests an element in the incremental update queue

Client becomes a receiver  
Client receives incremental push updates  
Client closes the transport queue tq  
Client closes the connection

Figure 3: ALTO New Transport Information Structure.

4. Transport Queue

4.1. Transport Queue Operations

A transport queue supports three basic operations (CRD): create, read (get status), and delete.

Create a transport queue: An ALTO client creates a transport queue using the HTTP POST method with ALTO SSE AddUpdateReq ([RFC 8895] Sec. 6.5) as the parameter:

```json
object {
    [ResourceID resource-id;]
    [JSONString tag;]
    [Boolean incremental-changes;]
    [Object input;]
} AddUpdateReq;
```

A successful POST request MUST return the URI for the transport queue. Unless the request has incremental-changes to be false, the client is added to receiver set as well, indicating that the client will receive automatic, incremental push updates.

Read a transport queue: A client reads the status of a transport queue by issuing a GET request to the transport queue URI returned from the POST method.
Delete a transport queue: a transport queue exposed to a client can be closed (deleted) either explicitly or implicitly.

* Explicit delete: A client uses the HTTP DELETE method to explicitly delete a transport queue. If successful, the transport queue is deleted from the local view of the client, although the server may still maintain the transport queue for other client connections.

* Implicit delete: Transport queue for a client is ephemeral: the close of the HTTP connection between the client and the server deletes the transport queue from the client’s view --- when the client reconnects, the client MUST NOT assume that the transport queue is still valid.

Error codes: ALTO/H2 uses HTTP error codes.

4.2. Examples

The first example is a client creating a transport queue.

Client -> server request

HEADERS
   - END_STREAM
+ END_HEADERS
 :method = POST
 :scheme = https
 :path = /tqs
 host = alto.example.com
 accept = application/alto-error+json,
            application/alto-transport+json
 content-type = application/alto-transport+json
 content-length = TBD

DATA
   - END_STREAM
   {
      "resource-id": "my-routingcost-map"
   }
Server -> client response:

HEADERS
- END_STREAM
+ END_HEADERS
:status = 200
content-type = application/alto-transport+json
content-length = TBD

DATA
- END_STREAM
{"tq": /tqs/2718281828459}

The client can then read the status of the transport queue using the read operation (GET) in the same HTTP connection. Below is an example (structure of incremental updates queue will be specified in the next section):
Client -> server request

HEADERS
- END_STREAM
+ END_HEADERS
  :method = GET
  :scheme = https
  :path = /tqs/2718281828459
  host = alto.example.com
  accept = application/alto-error+json,
           application/alto-transport+json

Server -> client response:

HEADERS
- END_STREAM
+ END_HEADERS
  :status = 200
  content-type = application/alto-transport+json
  content-length = TBD

DATA
- END_STREAM

{ "ug":
  [ {seq: 101,
     "media-type": "application/alto-costmap+json",
     tag: "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe" },
   {seq: 102,
     "media-type": "application/merge-patch+json",
     tag: "cdf0222x59740b0b2e3f8eb1d4785acd42231bfe" },
   {seq: 103,
     "media-type": "application/merge-patch+json",
     tag: "8eb1d4785acd42231bfecdf0222x59740b0b2e3f",
     "link": "/tqs/2718281828459/snapshot/2e3f" } ],
  "rs": ["self"]
}

5. Incremental Updates Queue
5.1. Incremental Updates Queue Operations

Among the CRUD operations, an incremental updates queue supports only the read operation: a client cannot create, update, or delete incremental updates queue directly---it is read only, and associated with transport queue automatically.

Reads an incremental updates queue: A client reads the status of an incremental updates queue using the HTTP GET method: GET transport-queue-uri/uq, where the transport-queue-uri is the URI returned in the transport queue create method.

The response informs the client the backlog status, and potential direct links. Specifically, the response is a JSON array, with each element being one incremental update, with three required fields and one optional field:

* "seq": a required JSON integer indicating the sequence number of the incremental update; As JSON allows a large integer space, when the server reaches the largest integer, the server SHOULD close the incremental update queue;

* "media-type", a required JSON string giving the type of the incremental update (see ALTO/SSE);

* "tag": a required JSON string giving a unique tag (see [RFC7285]);

* "link": an optional JSON string giving an optional link for a client to directly request a resource as a complete snapshot (not through incremental updates).

Note that the server determines the state (window of history and type of each update) in the incremental updates queue, as specified by [R4].

5.2. Examples

Assume the same example in the preceding section. The client can check the status of the incremental updates queue of a transport queue from the same connection:
Client -> server request:

HEADERS
   - END_STREAM
+ END_HEADERS
    :method = GET
    :scheme = https
    :path = /tgs/2718281828459/uq
    host = alto.example.com
    accept = application/alto-error+json,
             application/alto-transport+json

Server -> client response:

HEADERS
   - END_STREAM
+ END_HEADERS
    :status = 200
    content-type = application/alto-transport+json
    content-length = TBD

DATA
   - END_STREAM
{
   [  
      {seq: 101,
       "media-type": "application/alto-costmap+json",
       "tag": "a10ce8b059740b0b2e3f8eb1d4785acd42231bfe" },
      {seq: 102,
       "media-type": "application/merge-patch+json",
       tag: "cdf0222x59740b0b2e3f8eb1d4785acd42231bfe" },
      {seq: 103,
       "media-type": "application/merge-patch+json",
       tag: "8eb1d4785acd42231bfecedf0222x59740b0b2e3f",
       "link": "/tgs/2718281828459/snapshot/2e3f"}  
   ],
}

6. Individual Updates

6.1. Individual Updates Operations

A client can only read an individual update. The read can be either pull read issued by the client or a push from the server to the client.
Client pull read: A client uses HTTP GET method on the incremental updates queue concatenated by the "seq" to pull an individual update.

Server push read: a client starts to receive server push when it is added to the receiver set. A client can add itself to the receiver set when creating the transport queue, or add itself explicitly to the receiver set (see the next section).

The work flow of server push of individual updates is the following:

* Initialization: the first update pushed from the server to the client MUST be the later of the following two: (1) the last independent update in the incremental updates queue; and (2) the following entry of the entry that matches the tag when the client creates the transport queue. The client MUST set SETTINGS_ENABLE_PUSH to be consistent.

* Push state: the server MUST maintain the last entry pushed to the client (and hence per client, per connection state) and schedule next update push accordingly.

* Push management: The client MUST NOT cancel (RST_STREAM) a PUSH_PROMISE to avoid complex server state management.

6.2. Examples

The first example is a client pull example, in which the client directly requests an individual update.
Client -> server request:

HEADERS
  + END_STREAM
  + END_HEADERS
  :method = GET
  :scheme = https
  :path = /tqs/2718281828459/uq/101
  host = alto.example.com
  accept = application/alto-error+json,
            application/alto-costmap+json

Server -> client response:

HEADERS
  - END_STREAM
  + END_HEADERS
  :status = 200
  content-type = application/alto-costmap+json
  content-length = TBD

DATA
  + END_STREAM
  {
    "meta": {
      "dependent-vtags": [{
        "resource-id": "my-network-map",
        "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
      }],
      "cost-type": {
        "cost-mode": "numerical",
        "cost-metric": "routingcost"
      },
      "vtag": {
        "resource-id": "my-routingcost-map",
        "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
      }
    },
    "cost-map": {
      "PID1": { "PID1": 1, "PID2": 5, "PID3": 10 },
      "PID2": { "PID1": 5, "PID2": 1, "PID3": 15 },
      "PID3": { "PID1": 20, "PID2": 15 }
    }
  }

Note from the transport queue state that the 103 message has an
OPTIONAL link to a complete snapshot, which a client can request.
Instead of directly requesting, the client can wait for the server for incremental push, where the server first sends PUSH_PROMISE with the GET URI as above.

Server -> client PUSH_PROMISE in current stream:

| PUSH_PROMISE |
| - END_STREAM |
| Promised Stream 4 |
| HEADER BLOCK |
| :method = GET |
| :scheme = https |
| :path = /tqs/2718281828459/uq/101 |
| host = alto.example.com |
| accept = application/alto-error+json, application/alto-costmap+json |

Server -> client content Stream 4:

| HEADERS |
| + END_STREAM |
| + END_HEADERS |
| :status = 200 |
| content-type = application/alto-costmap+json |
| content-length = TBD |

| DATA |
| + END_STREAM |

```json
{
  "meta": {
    "dependent-vtags": [{
      "resource-id": "my-network-map",
      "tag": "da65eca2eb7a10ce8b059740b0b2e3f8eb1d4785"
    }],
    "cost-type": {
      "cost-mode": "numerical",
      "cost-metric": "routingcost"
    },
    "vtag": {
      "resource-id": "my-routingcost-map",
      "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
    }
  },
  "cost-map": {
    "PID1": { "PID1": 1, "PID2": 5, "PID3": 10 },
    "PID2": { "PID1": 5, "PID2": 1, "PID3": 15 },
    "PID3": { "PID1": 20, "PID2": 15 }
  }
}
```
Server -> client PUSH_PROMISE in current stream:

PUSH_PROMISE
  - END_STREAM
Promised Stream 6
HEADER BLOCK
  :method = GET
  :scheme = https
  :path = /tgs/2718281828459/uq/102
  host = alto.example.com
  accept = application/alto-error+json,
           application/merge-patch+json

Server -> client content Stream 6

HEADERS
  + END_STREAM
  + END_HEADERS
  :status = 200
  content-type = application/merge-patch+json
  content-length = TBD

DATA
  + END_STREAM
  { ... }

7. Receiver Set

7.1. Receiver Set Operations

Among the CRUD operations, a client can add to or delete itself from the receiver set of a transport queue. It can also read the status of the receiver set.

Create: A client can add itself in the receiver set by using the HTTP PUT method: DELETE transport-queue/rs/self

Read: A client can see only itself in the receiver set. The appearance of self in the receiver set (read does not return not exists) is an indication that push starts.

Delete: A client can delete itself (stops receiving push) either explicitly or implicitly.
* Explicit delete: A client deletes itself using the HTTP DELETE method: DELETE transport-queue/rs/self.

* Implicit delete: Transport queue is connection ephemeral: close of connection or stream for the transport queue deletes the transport queue (from the view) for the client.

7.2. Examples

A client can stop incremental push updates by sending the request:

DELETE /tqs/2718281828459/rs/self HTTP/2
Accept: application/alto-transport+json

HTTP/2 200 OK

8. ALTO/H2 Stream Management

8.1. Objectives

A main benefit of using HTTP/2 for ALTO is to take advantage of HTTP/2 streams. In particular, the objectives of ALTO/H2 include:

* Allow stream concurrency to reduce latency
* Minimize the number of streams created
* Enforce dependency among streams (so that if A depends on B, then A should be sent before B)
* Encode dependency to enforce semantics (correctness)

To realize the objectives specified in the preceding section, ALTO/H2 MUST satisfy the following stream management requirements in all 4 phases specified in the next 4 subsections.

8.2. Client -> Server [Create Transport Queue]

Each request to create a transport queue (POST) MUST choose a new client selected stream ID (SID_tq), with the following requirements:

* Stream Identifier of the frame is a new client-selected stream ID; Stream Dependency in HEADERS is 0 (connection) for an independent resource, the other transport queue if the dependency is known.
* Invariant: Stream keeps open until close or error.
8.3. Client -> Server [Close Transport Queue]

DELETE to close a transport queue (SID_tq) MUST be sent in SID_tq, with the following requirements:

* Stream Identifier of the frame is SID_tq, and Stream Dependency in HEADER is 0 (connection), so that a client cannot close a different stream.

* HEADERS indicates END_STREAM; server response SHOULD close the stream.

8.4. Client -> Server [Request on Data of a Transport Queue on Stream SID_tq]

The request and response MUST satisfy the following requirements:

* The Stream Identifier of the frame is a new client-selected stream ID, and Stream Dependency in HEADERS MUST be SID_tq, so that a client cannot issue request on a closed transport queue;

* Both the request and the response MUST indicate END_STREAM.

8.5. Server -> Client [PUSH_PROMISE for Transport Queue on Stream SID_tq]

The server push MUST satisfy the following requirements:

* PUSH_PROMISE MUST be sent in stream SID_tq to serialize to allow the client to know the push order;

* Each PUSH_PROMISE chooses a new server-selected stream ID, and the stream is closed after push.

8.6. Concurrency Management

* ALTO/H2 must allow concurrency control using the SETTINGS_MAX_CONCURRENT_STREAMS option in HTTP/2.

* From the client to the server direction, there MUST be one stream for each open transport queue, and hence a client can always close a transport queue (which it uses to open the stream) and hence can also close, without the risk of deadlock.

* From the server to the client direction, each push needs to open a new stream and this should be controlled by SETTINGS_MAX_CONCURRENT_STREAMS.
9. ALTO/H2 Information Resource Directory (IRD)

Extending the IRD example in Section 8.1 of [RFC8895], below is the IRD of an ALTO server supporting ALTO base protocol, ALTO/SSE, and ALTO/H2.

In particular,

```
"my-network-map": {
  "uri": "https://alto.example.com/networkmap",
  "media-type": "application/alto-networkmap+json",
},
"my-routingcost-map": {
  "uri": "https://alto.example.com/costmap/routingcost",
  "media-type": "application/alto-costmap+json",
  "uses": ["my-networkmap"],
  "capabilities": {
    "cost-type-names": ["num-routingcost"]
  }
},
"my-hopcount-map": {
  "uri": "https://alto.example.com/costmap/hopcount",
  "media-type": "application/alto-costmap+json",
  "uses": ["my-networkmap"],
  "capabilities": {
    "cost-type-names": ["num-hopcount"]
  }
},
"my-filtered-cost-map": {
  "uri": "https://alto.example.com/costmap/filtered/constraints",
  "media-type": "application/alto-costmap+json",
  "accepts": "application/alto-costmapfilter+json",
  "uses": ["my-networkmap"],
  "capabilities": {
    "cost-type-names": ["num-routingcost", "num-hopcount"],
    "cost-constraints": true
  }
},
"my-simple-filtered-cost-map": {
  "uri": "https://alto.example.com/costmap/filtered/simple",
  "media-type": "application/alto-costmap+json",
  "accepts": "application/alto-costmapfilter+json",
  "uses": ["my-networkmap"],
  "capabilities": {
    "cost-type-names": ["num-routingcost", "num-hopcount"],
    "cost-constraints": false
  }
},
```
"my-props": {
  "uri": "https://alto.example.com/properties",
  "media-type": "application/alto-endpointprops+json",
  "accepts": "application/alto-endpointpropparams+json",
  "capabilities": {
    "prop-types": ["priv:ietf-bandwidth"]
  }
},
"my-pv": {
  "uri": "https://alto.example.com/endpointcost/pv",
  "media-type": "multipart/related;
              type=application/alto-endpointcost+json",
  "accepts": "application/alto-endpointcostparams+json",
  "capabilities": {
    "cost-type-names": [ "path-vector" ],
    "ane-properties": [ "maxresbw", "persistent-entities" ]
  }
},
"update-my-costs": {
  "uri": "https://alto.example.com/updates/costs",
  "media-type": "text/event-stream",
  "accepts": "application/alto-updatestreamparams+json",
  "uses": [
    "my-network-map",
    "my-routingcost-map",
    "my-hopcount-map",
    "my-simple-filtered-cost-map"
  ],
  "capabilities": {
    "incremental-change-media-types": {
      "my-network-map": "application/json-patch+json",
      "my-routingcost-map": "application/merge-patch+json",
      "my-hopcount-map": "application/merge-patch+json"
    },
    "support-stream-control": true
  }
},
"update-my-costs-h2": {
  "uri": "https://alto.example.com/updates-h2/costs",
  "media-type": "application/alto-h2",
  "accepts": "application/alto-updatestreamparams+json",
  "uses": [
    "my-network-map",
    "my-routingcost-map",
    "my-hopcount-map",
    "my-simple-filtered-cost-map"
  ],
  "capabilities": {
"incremental-change-media-types": {
    "my-network-map": "application/json-patch+json",
    "my-routingcost-map": "application/merge-patch+json",
    "my-hopcount-map": "application/merge-patch+json"
},
"support-stream-control": true
},

"update-my-props": {
    "uri": "https://alto.example.com/updates/properties",
    "media-type": "text/event-stream",
    "uses": [ "my-props" ],
    "accepts": "application/alto-updatestreamparams+json",
    "capabilities": {
        "incremental-change-media-types": {
            "my-props": "application/merge-patch+json"
        },
        "support-stream-control": true
    }
},

"update-my-pv": {
    "uri": "https://alto.example.com/updates/pv",
    "media-type": "text/event-stream",
    "uses": [ "my-pv" ],
    "accepts": "application/alto-updatestreamparams+json",
    "capabilities": {
        "incremental-change-media-types": {
            "my-pv": "application/merge-patch+json"
        },
        "support-stream-control": true
    }
}

Note that it is straightforward for an ALTO server to run HTTP/2 and support concurrent retrieval of multiple resources such as "my-network-map" and "my-routingcost-map" using multiple HTTP/2 streams with the need to introducing ALTO/H2.

The resource "update-my-costs-h2" provides an ALTO/H2 based connection, and this is indicated by the media-type "application/alto-h2". For an ALTO/H2 connection, the client can send in a sequence of control requests using media type application/alto-updatestreamparams+json. The server creates HTTP/2 streams and pushes updates to the client.
10. Security Considerations

The properties defined in this document present no security considerations beyond those in Section 15 of the base ALTO specification [RFC7285].

11. IANA Considerations

IANA will need to register the alto-h2 media type under ALTO registry as defined in [RFC7285].

12. Acknowledgments

The authors of this document would also like to thank many for the reviews and comments.

13. References

13.1. Normative References


13.2. Informative References


Appendix A. Outlook to ALTO with HTTP/3

This draft is focusing on HTTP/2 enhancement of the ALTO protocol and the design takes advantage of HTTP/2 design features such as parallel transfer and respects HTTP/2 semantics (e.g., PUSH_PROMISE). Since QUIC and HTTP/3 respectively are coming up for various protocols on the Internet it is understandable that the question arises, if ATLO could also take advantage of the advantages of HTTP/3. QUIC can be seen as a replacement for TCP+TLS+HTTP2. HTTP/3 bases on the QUIC transport protocol and uses UDP instead of a TCP connection.

QUIC has been developed by the IETF QUIC Working Group with the following goals:

* Minimizing connection establishment and overall transport latency for applications, starting with HTTP/2
* Providing multiplexing without head-of-line blocking
* Requiring only changes to path endpoints to enable deployment
* Enabling multipath and forward error correction extensions
* Providing always-secure transport, using TLS 1.3 by default

If HTTP/3 is not supported, it automatically runs on HTTP/2. The prerequisite for HTTP/3 is that both client and server support it.

The basic assumption is that an implementation that runs on HTTP/2 should also run-on HTTP/3. This should be transparent. HTTP/3 uses "well known port" UDP 443 analogous to TCP 443. The network between client and server must not filter HTTP/3.
Since many applications still using HTTP/2 it is mandatory for ALTO to support this protocol first. This ensures compatibility. Therefore, this document describes the update of ALTO from HTTP/1.x to HTTP/2. The usage of HTTP/3 will be described in a separate document so that compatibility of ALTO with HTTP/3 will be ensured in a later stage.

Authors’ Addresses

Roland Schott
Deutsche Telekom
Heinrich-Hertz-Strasse 3-7
64295 Darmstadt
Germany
Email: Roland.Schott@telekom.de

Y. Richard Yang
Yale University
51 Prospect St
New Haven, CT 06520
United States of America
Email: yry@cs.yale.edu

Kai Gao
Sichuan University
Chengdu
201804
China
Email: kgao@scu.edu.cn

Jingxuan Jensen Zhang
Tongji University
4800 Cao’An Hwy
Shanghai
201804
China
Email: jingxuan.n.zhang@gmail.com
A Yang Data Model for OAM and Management of ALTO Protocol
draft-zhang-alto-oam-yang-02

Abstract

This document defines a YANG data model for Operations, Administration, and Maintenance (OAM) & Management of Application-Layer Traffic Optimization (ALTO) Protocol. The operator can use the data model to create and update ALTO information resources, manage the access control, configure server-to-server communication and server discovery, and collect statistical data.

Discussion Venues

This note is to be removed before publishing as an RFC.

Discussion of this document takes place on the ALTO Working Group mailing list (alto@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/alto/.

Source for this draft and an issue tracker can be found at https://github.com/openalto/draft-alto-oam-yang.

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 https://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 8 September 2022.

Copyright Notice

Copyright (c) 2022 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
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 Revised BSD License text as
described in Section 4.e of the Trust Legal Provisions and are
provided without warranty as described in the Revised BSD License.

Table of Contents

1. Introduction ..................................................... 3
2. Requirements Language ........................................... 3
3. Terminology ..................................................... 3
   3.1. Tree Diagrams .............................................. 3
   3.2. Prefixes in Data Node Names ............................... 4
4. Design Scope and Requirements ................................. 4
   4.1. Scope of Data Model for ALTO O&M ......................... 4
   4.2. Basic Requirements ....................................... 5
   4.3. Additional Requirements for Extensibility ............... 6
5. Design of ALTO O&M Data Model ................................. 6
   5.1. Overview of ALTO O&M Data Model .......................... 6
   5.2. Meta Information of ALTO Server ........................... 8
   5.3. ALTO Information Resources Configuration Management ... 8
   5.4. Data Sources .............................................. 11
      5.4.1. Yang DataStore Data Source .......................... 12
      5.4.2. Prometheus Data Source ............................... 12
   5.5. Model for ALTO Server-to-server Communication .......... 12
6. Design of ALTO O&M Statistics Data Model ..................... 13
7. Extension of ALTO O&M Data Model ............................... 14
8. ALTO OAM YANG Module ........................................... 15
   8.1. The ietf-alto Module ..................................... 15
   8.2. The ietf-alto-stats Module ............................... 27
9. Security Considerations ......................................... 30
10. IANA Considerations ........................................... 30
11. References .................................................... 31
   11.1. Normative References .................................... 31
   11.2. Informative References .................................. 32
Appendix A. Example Module for Information Resource Creation
   Algorithm ..................................................... 33
Acknowledgements .................................................. 34
1. Introduction

This document defines a YANG data model for the Operations, Administration, and Maintenance (OAM) & Management of Application-Layer Traffic Optimization (ALTO) Protocol. The basic purpose of this YANG data model is discussed in Section 16 of [RFC7285]. The operator can use the data model to create and update ALTO information resources, manage the access control, configure server-to-server communication and server discovery, and collect statistical data.

The basic structure of this YANG data model is guided by Section 16 of [RFC7285] and [RFC7971]. Although the scope of the YANG data model in this document mainly focuses on the support of the base ALTO protocol [RFC7285] and the existing ALTO standard extensions (including [RFC8189], [RFC8895] and [RFC8896]), the design will also be extensible for future standard extensions (e.g., [I-D.ietf-alto-path-vector], [I-D.ietf-alto-unified-props-new], [I-D.ietf-alto-cdni-request-routing-alto], and [I-D.ietf-alto-performance-metrics]).

2. Requirements Language

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here. When the words appear in lower case, they are to be interpreted with their natural language meanings.

3. Terminology

This document uses the following acronyms:

* OAM - Operations, Administration, and Maintainance

* O&M - OAM and Management

3.1. Tree Diagrams

A simplified graphical representation of the data model is used in this document. The meaning of the symbols in these diagrams is defined in [RFC8340].
3.2. Prefixes in Data Node Names

In this document, names of data nodes and other data model objects are often used without a prefix, as long as it is clear from the context in which YANG module each name is defined. Otherwise, names are prefixed using the standard prefix associated with the corresponding YANG module, as shown in Table 1.

<table>
<thead>
<tr>
<th>Prefix</th>
<th>YANG module</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>yang</td>
<td>ietf-yang-types</td>
<td>[RFC6991]</td>
</tr>
<tr>
<td>inet</td>
<td>ietf-inet-types</td>
<td>[RFC6991]</td>
</tr>
<tr>
<td>key-chain</td>
<td>ietf-key-chain</td>
<td>[RFC8177]</td>
</tr>
</tbody>
</table>

Table 1: Prefixes and corresponding YANG modules

4. Design Scope and Requirements

4.1. Scope of Data Model for ALTO O&M

What is in the scope of this document?

* Data model for deploy an ALTO server/client.
* Data model for operate and manage a running ALTO server/client.
* Data model for functionality/capability configuration for ALTO services.
* Data model for monitoring ALTO-related performance metrics.

What is not in the scope of this document?

This document does not define any data model related to specific implementation, including:

* Data structures for how to store/deliver ALTO information resources (e.g., database schema to store a network map).
* Data structures for how to store information collected from data sources. (e.g., database schema to store topology collected from an Interface to the Routing System (I2RS) client [RFC7921])
4.2. Basic Requirements

Based on discussions and recommendations in [RFC7285] and [RFC7971], the data model provided by this document satisfies basic requirements listed in Table 2.

<table>
<thead>
<tr>
<th>Requirement</th>
<th>Reference</th>
</tr>
</thead>
<tbody>
<tr>
<td>R1: The data model should support configuration for ALTO server setup.</td>
<td>Section 16.1 of [RFC7285]</td>
</tr>
<tr>
<td>R2: The data model should provide logging management.</td>
<td>Section 16.2.1 of [RFC7285]</td>
</tr>
<tr>
<td>R3: The data model should provide ALTO-related management information.</td>
<td>Section 16.2.2 of [RFC7285]</td>
</tr>
<tr>
<td>R4: The data model should provide metrics for server failures.</td>
<td>Section 16.2.3 of [RFC7285], Section 3.3 of [RFC7971]</td>
</tr>
<tr>
<td>R5-1: The data model should support configuration for different data sources.</td>
<td>Section 16.2.4 of [RFC7285], Section 3.2 of [RFC7971]</td>
</tr>
<tr>
<td>R5-2: The data model should support configuration for information resource generation algorithms.</td>
<td>Section 16.2.4 of [RFC7285]</td>
</tr>
<tr>
<td>R5-3: The data model should support configuration for access control at information resource level.</td>
<td>Section 16.2.4 of [RFC7285]</td>
</tr>
<tr>
<td>R6: The data model should provide performance monitoring for ALTO-specific metrics.</td>
<td>Section 16.2.5 of [RFC7285], Section 3.4 of [RFC7971]</td>
</tr>
<tr>
<td>R7: The data model should support configuration for security policy management.</td>
<td>Section 16.2.6 of [RFC7285]</td>
</tr>
</tbody>
</table>

Table 2: Basic Requirements of Data Model for ALTO O&M.
4.3. Additional Requirements for Extensibility

R8: As the ALTO protocol is extensible, the data model for ALTO O&M should allow for augmentation to support potential future extensions.

5. Design of ALTO O&M Data Model

5.1. Overview of ALTO O&M Data Model

The ietf-alto module defined in this document provide all the basic ALTO O&M data models fitting the requirements listed in Section 4.

The container "alto-server" in the ietf-alto module contains all the configured and operational parameters of the administrated ALTO server instance.

NOTE: So far, the ALTO YANG module only focuses on the ALTO server related configuration. The ALTO client related configuration will be added in a future version of the document.

module: ietf-alto
  +-rw alto-server
    |  +-rw hostname?   inet:host
    |  +-rw cost-type*  [cost-type-name]
    |     |  +-rw cost-type-name  string
    |     |  +-rw cost-mode       cost-mode
    |     |  +-rw cost-metric     cost-metric
    |  +-rw meta* [meta-key]
    |     |  +-rw meta-key        string
    |     |  +-rw meta-value      string
    |  +-rw resource* [resource-id]
    |     |  +-rw resource-id     resource-id
    |     |  +-rw resource-type   identityref
    |     |  +-rw description?    string
    |     |  +-rw accepted-group* string
    |     |  +-rw dependency*     resource-id
    |  +-rw auth
    |     |  +-rw (auth-type-selection)?
    |     |      |  +-:(auth-key-chain)
    |     |      |     |  +-rw key-chain?   key-chain:key-chain-ref
    |     |      |     |  +-:(auth-key)
    |     |      |     |  +-:(auth-tls)
    |  +-rw (resource-params)?
    |     |  +-:(ird)
    |     |     |  +-rw alto-ird-params
    |     |     |     |  +-rw delegation   inet:uri
    |     |  +-:(networkmap)
    |     |     |  +-rw alto-networkmap-params
++-rw is-default?  boolean
++-rw filtered?  boolean
++-rw (algorithm)
++-:(costmap)
  +++-rw alto-costmap-params
  +--rw filtered?  boolean
  +--rw cost-type-names*  string
  +--rw cost-constraints?  boolean
  +--rw max-cost-types?  uint32 {multi-cost}?  
  +--rw testable-cost-type-names*  string {multi-cost}?  
  +++-rw calendar-attributes {cost-calendar}?  
    +--rw cost-type-names*  string
    +--rw time-interval-size  decimal64
    +--rw number-of-intervals  uint32
  +++-rw (algorithm)
++-:(endpointcost)
  +++-rw alto-endpointcost-params
  +--rw cost-type-names*  string
  +--rw cost-constraints?  boolean
  +--rw max-cost-types?  uint32 {multi-cost}?  
  +--rw testable-cost-type-names*  string {multi-cost}?  
  +++-rw calendar-attributes {cost-calendar}?  
    +--rw cost-type-names*  string
    +--rw time-interval-size  decimal64
    +--rw number-of-intervals  uint32
  +++-rw (algorithm)
++-:(endpointprop)
  +++-rw alto-endpointprop-params
  +--rw prop-types*  string
  +--rw (algorithm)
++-:(propmap)  {propmap}?  
  +++-rw alto-propmap-params
  +--rw (algorithm)
++-:(cdni)  {cdni}?  
  +++-rw alto-cdni-params
  +--rw (algorithm)
++-:(update)  {incr-update}?  
  +++-rw alto-update-params
  +--rw (algorithm)
++-rw data-source*  [source-id]
  +--rw source-id  string
  +--rw source-type  identityref
  +--rw (update-policy)
    +--:(reactive)
      +--rw reactive  boolean
    +--:(proactive)
      +--rw poll-interval  uint32
  +--rw (source-params)?
5.2. Meta Information of ALTO Server

The ALTO server instance contains the following basic configurations for the server setup.

The hostname is the name that is used to access the ALTO server. It will be also used in the URI of each information resource provided by the ALTO server.

The cost type list is the registry for the cost types that can be used in the ALTO server.

The meta list contains the customized meta data of the ALTO server. It will be populated into the meta field of the default Information Resource Directory (IRD).

TODO: As suggested by [RFC7286] and [RFC8686], the configuration related to ALTO server discovery should also be included here.

5.3. ALTO Information Resources Configuration Management

The ALTO server instance contains a list of resource entries. Each resource entry contains the configurations of an ALTO information resource (See Section 8.1 of [RFC7285]). The operator of the ALTO server can use this model to create, update, and remove the ALTO information resource.
Each resource entry provides configuration defining how to create or update an ALTO information resource. Adding a new resource entry will submit an ALTO information resource creation intent to the intent system to create a new ALTO information resource. Updating an existing resource entry will update the corresponding ALTO information resource creation intent. Removing an existing resource entry will remove the corresponding ALTO information resource creation intent and also the created ALTO information resource.

The parameter of the intent interface defined by a resource entry MUST include a unique resource-id and a resource-type.

It can also include an accepted-group node containing a list of user-groups that can access this ALTO information resource.

As section 15.5.2 of [RFC7285] suggests, the module also defines authentication-related configuration to employ access control at information resource level. The ALTO server returns the IRD to the ALTO client based on its authentication information.

For some resource-type, the parameter of the intent interface MUST also include the dependency node containing the resource-id of the dependent ALTO information resources (See Section 9.1.5 of [RFC7285]).

For each type of ALTO information resource, the creation intent MAY also need type-specific parameters. These type-specific parameters include two categories:

1. One category of the type-specific parameters are common for the same type of ALTO information resource. They declare the Capabilities of the ALTO information resource (See Section 9.1.3 of [RFC7285]).

2. The other categories of the type-specific parameters are algorithm-specific. The developer of the ALTO server can implement their own creation algorithms and augment the algorithm node to declare algorithm-specific input parameters.

Except for the ird resource, all the other types of resource entries have augmented algorithm node. The augmented algorithm node can reference data sources subscribed by the data-source entries (See Section 5.4).
The developer cannot customize the creation algorithm of the ird resource. The default ird resource will be created automatically based on all the added resource entries. The delegated ird resource will be created as a static ALTO information resource (See Section 9.2.4 of [RFC7285]).

module: ietf-alto
+++rw alto-server
...
+++rw resource* [resource-id] resource-id
  +++rw resource-id
  +++rw resource-type identityref
  +++rw description? string
  +++rw accepted-group* string
  +++rw dependency* resource-id
  +++rw auth
    +++rw (auth-type-selection)?
      +++rw (auth-key-chain)
        +++rw key-chain? key-chain:key-chain-ref
        +++rw (auth-key)
        +++rw (auth-tls)
    +++rw (resource-params)?
      +++rw alto-ird-params
        +++rw delegation inet:uri
      +++rw (networkmap)
        +++rw alto-networkmap-params
          +++rw is-default? boolean
          +++rw filtered? boolean
          +++rw (algorithm)
      +++rw (costmap)
        +++rw alto-costmap-params
          +++rw filtered? boolean
          +++rw cost-type-names* string
          +++rw cost-constraints? boolean
          +++rw max-cost-types? uint32 {multi-cost}?
          +++rw testable-cost-type-names* string {multi-cost}?
          +++rw calendar-attributes {cost-calendar}?
            +++rw cost-type-names* string
            +++rw time-interval-size decimal64
            +++rw number-of-intervals uint32
            +++rw (algorithm)
      +++rw (endpointcost)
        +++rw alto-endpointcost-params
          +++rw cost-type-names* string
          +++rw cost-constraints? boolean
          +++rw max-cost-types? uint32 {multi-cost}?
          +++rw testable-cost-type-names* string {multi-cost}?
5.4. Data Sources

The ALTO server instance contains a list of data-source entries to subscribe the data sources from which ALTO information resources are derived (See Section 16.2.4 of [RFC7285]).

A data-source entry MUST include:

* a unique source-id for resource creation algorithms to reference,

* the source-type attribute to declare the type of the data source,

* the update-policy to specify how to get the data update from the data source,

* the source-params to specify where and how to query the data.

The update policy can be either reactive or proactive. For the reactive update, the ALTO server gets the update as soon as the data source changes. For the proactive update, the ALTO server has to proactively fetch the data source periodically.

To use the reactive update, the reactive attribute MUST be set true. To use the proactive update, the poll-interval attribute MUST be greater than zero. The value of poll-interval specifies the interval of fetching the data in milliseconds. If reactive is false or poll-interval is zero, the ALTO server will not update the data source.
The data-source/source-params node can be augmented for different types of data sources. This data model only includes import interfaces for a list of predefined data sources. More data sources can be supported by future documents and other third-party providers.

module: ietf-alto
  +--rw alto-server
    ...
    +--rw data-source* [source-id]
      +--rw source-id string
      +--rw source-type identityref
      +--rw (update-policy)
        |   +--:(reactive)
        |     +--rw reactive boolean
        |   +--:(proactive)
        |       +--rw poll-interval uint32
      +--rw (source-params)?
        +--:(yang-datastore)
          +--rw yang-datastore-source-params
            +--rw source-path yang:xpath1.0
        +--:(prometheus)
          +--rw prometheus-source-params
            +--rw source-uri inet:uri
            +--rw query-data? string

Note: Current source configuration still has limitations. It should be revised to support more general southbound and data retrieval mechanisms.

5.4.1. Yang DataStore Data Source

The yang-datastore-source-params is used to import the YANG data which is located in the same YANG model-driven data store supplying the current ALTO O&M data model. The source-path is used to specify the XPath of the data source node.

5.4.2. Prometheus Data Source

The prometheus-source-params is used to import common performance metrics data which is provided by a Prometheus server. The source-uri is used to establish the connection with the Prometheus server. The query-data is used to specify the potential query expression in PromQL.

5.5. Model for ALTO Server-to-server Communication

TBD.

Zhang, et al. Expires 8 September 2022
6. Design of ALTO O&M Statistics Data Model

As section 16.2.5 of [RFC7285] suggests, the YANG data module defined in this document also contains statistics for ALTO-specific performance metrics.

More specifically, this data model contains the following measurement information suggested by [RFC7971]:

* Measurement of impact
  - Total amount and distribution of traffic
  - Application performance

* System and service performance
  - Requests and responses for each information resource
  - CPU and memory utilization
  - ALTO map updates
  - Number of PIDs
  - ALTO map sizes

Besides the measurement information suggested by [RFC7971], this data model also contains useful measurement information for other ALTO extensions:

* Number of generic ALTO entities (for [I-D.ietf-alto-unified-props-new] and [I-D.ietf-alto-cdni-request-routing-alto])

* Statistics for update sessions and events (for [RFC8189])

* Statistics for calendar (for [RFC8896])

The module, "ietf-alto-stats", augments the ietf-alto module to include statistics at the ALTO server and information resource level.
module: ietf-alto-stats

augment /alto:alto-server/alto:resource:
    +++-ro num-res-upd?    yang:counter32
    +++-ro res-mem-size?   yang:counter32
    +++-ro res-enc-size?   yang:counter32

    /alto:networkmap/alto:alto-networkmap-params:
        +++-ro num-map-pid?    yang:counter32

    /alto:propmap/alto:alto-propmap-params:
        +++-ro num-map-entry?  yang:counter32

    /alto:cdni/alto:alto-cdni-params:
        +++-ro num-base-obj?   yang:counter32

    /alto:update/alto:alto-update-params:
        +++-ro num-upd-sess     yang:counter32
        +++-ro num-event-total  yang:counter32
        +++-ro num-event-max?   yang:counter32
        +++-ro num-event-min?   yang:counter32
        +++-ro num-event-avg?   yang:counter32

7. Extension of ALTO O&M Data Model

As ALTO protocol is extensible, new protocol extensions can be
developed after this data model is published. To support future ALTO
protocol extensions, the extension documents can augment the existing
cases of the resource-params choice with new configuration parameters
for existing ALTO information resource extensions, or augment the
resource-params with new cases for new ALTO information resources.

Developers and operators can also extend this ALTO O&M data model to
align with their own implementations. Specifically, the following
nodes of the data model can be augmented:

* The algorithm choice of the resource-params of each resource.

* The data-source choice.

The following example shows how the developer augments the algorithm
choice of alto-networkmap-params with a creation algorithm for the
network map resource.

module: example-ietf-alto-alg

    /alto:networkmap/alto:alto-networkmap-params
    /alto:algorithm:
        +---:(l3-unicast-cluster)
            +--rw l3-unicast-cluster-algorithm
                +--rw l3-unicast-topo
                    |         -> /alto:alto-server/data-source/source-id
                +--rw depth?             uint32

This example defines a creation algorithm called l3-unicast-cluster-algorithm for the network map resource. It takes two algorithm-specific parameters:

l3-unicast-topo This parameter refers to the source id of a data source node subscribed in the data-source list (See Section 5.4). The corresponding data source is assumed to be an internal data source (See Section 5.4.1) for an IETF layer 3 unicast topology defined in [RFC8346]. The algorithm uses the topology data from this data source to compute the ALTO network map resource.

depth This optional parameter sets the depth of the clustering algorithm. For example, if the depth sets to 1, the algorithm will generate PID for every l3-node in the topology.

The creation algorithm can be reactively called once the referenced data source updates. Therefore, the ALTO network map resource can be updated dynamically. The update of the reference data source depends on the used update-policy (See Section 5.4).

8. ALTO OAM YANG Module

8.1. The ietf-alto Module

<CODE BEGINS> file "ietf-alto@2022-03-07.yang"
module ietf-alto {
    yang-version 1.1;
    namespace
        "urn:ietf:params:xml:ns:yang:ietf-alto";
    prefix "alto";

    import ietf-inet-types {
        prefix "inet";
        reference
            "RFC 6991: Common YANG Data Types";
    }

import ietf-yang-types {
  prefix "yang";
  reference
    "RFC 6991: Common YANG Data Types";
}

import ietf-key-chain {
  prefix key-chain;
  reference
    "RFC 8177: YANG Data Model for Key Chains";
}

organization
  "IETF ALTO Working Group";

contact
  "WG Web:  <https://datatracker.ietf.org/wg/alto/about/>
           WG List:  <alto@ietf.org>";

description
  "This YANG module defines all the configured and operational
   parameters of the administrated ALTO server instance.

  Copyright (c) 2022 IETF Trust and the persons identified as
  authors of the code. All rights reserved.

  Redistribution and use in source and binary forms, with or
  without modification, is permitted pursuant to, and subject to
  the license terms contained in, the Revised BSD License set
  forth in Section 4.c of the IETF Trust's Legal Provisions
  Relating to IETF Documents

  This version of this YANG module is part of RFC XXXX
  (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself
  for full legal notices.

  The key words 'MUST', 'MUST NOT', 'REQUIRED', 'SHALL', 'SHALL
  NOT', 'SHOULD', 'SHOULD NOT', 'RECOMMENDED', 'NOT RECOMMENDED',
  'MAY', and 'OPTIONAL' in this document are to be interpreted as
  described in BCP 14 (RFC 2119) (RFC 8174) when, and only when,
  they appear in all capitals, as shown here.";

revision "2022-03-07" {
  description
    "Initial Version.";
  reference
    "RFC XXXX: A YANG Data Model for OAM and Management of ALTO"
typedef cost-mode {
  type enumeration {
    enum numerical {
      value 1;
      description
        "This mode indicates that it is safe to perform numerical
         operations";
    }
    enum ordinal {
      value 2;
      description
        "This mode indicates that the cost values in a cost map
         represent ranking";
    }
  }
  description
    "The cost mode attribute indicates how costs should be
     interpreted. Specifically, the cost mode attribute indicates
     whether returned costs should be interpreted as numerical
     values or ordinal rankings."
  reference
    "Section 6.1.2 of RFC 7285."
}

typedef cost-metric {
  type string;
  description
    "The cost metric attribute indicates what the cost
     represents."
  reference
    "Section 6.1.1 of RFC 7285."
}

typedef resource-id {
  type string {
    length "1..64";
    pattern "[0-9a-zA-Z\-:@_]*";
  }
  description
    "Format of Resource ID"
  reference
    "Section 9.1.1 of RFC 7285."
}

identity resource-types {

description
   "Base identity for type of information resource.";
}

identity source-types {
   description
      "Base identity for type of data source.";
}

identity network-map {
   base resource-types;
   description
      "Identity for network map.";
}

identity cost-map {
   base resource-types;
   description
      "Identity for cost map.";
}

identity property-map {
   base resource-types;
   description
      "Identity for property map.";
}

identity yang-datastore {
   base resource-types;
   description
      "Identity for data source of YANG-based datastore.";
}

identity prometheus {
   base resource-types;
   description
      "Identity for data source of prometheus system.";
}

feature multi-cost {
   description
      "Support multi-cost extension."
   reference
      "RFC 8189: Multi-Cost Application-Layer Traffic Optimization (ALTO)"
}

feature cost-calendar {

Zhang, et al. Expires 8 September 2022 [Page 18]
description
"Support cost calendar extension.";
reference
"RFC 8896: Application-Layer Traffic Optimization (ALTO) Cost Calendar";
}

feature propmap {
   description
   "Support entity property map extension.";
   reference
   "RFC XXXX: An ALTO Extension: Entity Property Maps";
}

feature cdni {
   description
   "Support CDNi extension.";
   reference
   "RFC XXXX: Content Delivery Network Interconnection (CDNI) Request Routing: CDNI Footprint and Capabilities Advertisement using ALTO";
}

feature incr-update {
   description
   "Support incremental update extension.";
   reference
   "RFC 8896: Application-Layer Traffic Optimization (ALTO) Incremental Updates Using Server-Sent Events (SSE)";
}

grouping filt-costmap-cap {
   description
   "This grouping defines data model for FilteredCostMapCapabilities.";
   reference
   "Sec 11.3.2.4 of RFC 7285.";
   leaf-list cost-type-names {
      type string;
      min-elements 1;
      description
      "Supported cost types";
   }
   leaf cost-constraints {
      type boolean;
      description
      "If true, then the ALTO server allows cost constraints to be included in requests to the"
corresponding URI. If not present, this field MUST be interpreted as if it specified false.

leaf max-cost-types {
  if-feature "multi-cost";
  type uint32;
  default 0;
  description
  "If present with value N greater than 0, this resource understands the multi-cost extensions in this document and can return a multi-cost map with any combination of N or fewer cost types in the 'cost-type-names' list. If omitted, the default value is 0.";
}

leaf-list testable-cost-type-names {
  if-feature "multi-cost";
  type string;
  description
  "If present, the resource allows constraint tests, but only on the cost type names in this array.";
}

container calendar-attributes {
  if-feature "cost-calendar";
  leaf-list cost-type-names {
    type string;
    min-elements 1;
    description
    "An array of one or more elements indicating the cost type names in the IRD entry to which the values of 'time-interval-size' and 'number-of-intervals' apply.";
  }
  leaf time-interval-size {
    type decimal64 {
      fraction-digits 4;
    }
    mandatory true;
    description
    "The duration of an ALTO Calendar time interval in a unit of seconds.";
  }
  leaf number-of-intervals {
    type uint32 {
      range "1..max";
    }
    mandatory true;
    description
    "A strictly positive integer (greater or equal to 1) that indicates the number of values of the Cost Calendar";"
array."
);

description
  "Configuration for CalendarAttributes.";
reference
  "Section 4.1 of RFC 8896.";
}

}

grouping algorithm {
  choice algorithm {
    mandatory true;
    description
      "Information resource creation algorithm to be augmented.";
  }
  description
    "This grouping defines base data model for information resource creation algorithm.";
}

container alto-server {
  description
    "The ALTO server instance.";
  leaf hostname {
    type inet:host;
    description
      "The name that is used to access the ALTO server.";
  }
  list cost-type {
    key "cost-type-name";
    leaf cost-type-name {
      type string;
      description
        "The name to reference cost type";
    }
    leaf cost-mode {
      type cost-mode;
      mandatory true;
      description
        "The referenced cost mode";
    }
    leaf cost-metric {
      type cost-metric;
      mandatory true;
      description
        "The referenced cost metric";
    }
    description

"Mapping between name and referenced cost type";
}
list meta {
key "meta-key";
leaf meta-key {
  type string;
  description
    "Custom meta key";
}
leaf meta-value {
  type string;
  mandatory true;
  description
    "Custom meta value";
} 

description
  "Mapping of custom meta information";
reference
  "Section 8.4.1 of RFC 7285.";
}
list resource {
key "resource-id";
leaf resource-id {
  type resource-id;
  description
    "resource-id to be defined.";
}
leaf resource-type {
  type identityref {
    base resource-types;
  } 
  mandatory true;
  description
    "identityref to be defined.";
}
leaf description {
  type string;
  description
    "The optional description for this information resource.";
}
leaf-list accepted-group {
  type string;
  description
    "Access list for authenticated clients.";
}
leaf-list dependency {
  type resource-id;
  description

"A list of dependent information resources."
}

container auth {
  description
  "The authentication options";
  choice auth-type-selection {
    description
    "Options for expressing authentication setting.”;
    case auth-key-chain {
      leaf key-chain {
        type key-chain:key-chain-ref;
        description
        "key-chain name.”;
      }
    }
    case auth-key {
    }
    case auth-tls {
    }
  }
}

choice resource-params {
  description
  "Resource-specific configuration.”;
  case ird {
    container alto-ird-params {
      leaf delegation {
        type inet:uri;
        mandatory true;
        description
        "Upstream IRD to be delegated”;
      }
      description
      "IRD-specific configuration”;
    }
  }
  case networkmap {
    container alto-networkmap-params {
      description
      "(Filtered) Network Map specific configuration”;
      reference
      "Section 11.2.1 and Section 11.3.1 of RFC 7285.”;
      leaf is-default {
        type boolean;
        description
        "Set whether this is the default network map”;
      }
    }
  }
leaf filtered {
    type boolean;
    default false;
    description
        "Configure whether filtered network map is supported."
}
uses algorithm;
}

case costmap {
    container alto-costmap-params {
        description
            "(Filtered) Cost Map specific configuration"
        reference
            "Section 11.2.2 and Section 11.3.2 of RFC 7285."
        leaf filtered {
            type boolean;
            description
                "Configure whether filtered cost map is supported."
        }
        uses filt-costmap-cap;
        uses algorithm;
    }
}

case endpointcost {
    container alto-endpointcost-params {
        description
            "Endpoint Cost Service specific configuration"
        reference
            "Section 11.5 of RFC 7285"
        uses filt-costmap-cap;
        uses algorithm;
    }
}

case endpointprop {
    container alto-endpointprop-params {
        description
            "Endpoint Cost Service specific configuration"
        reference
            "Section 11.5 of RFC 7285"
        leaf-list prop-types {
            type string;
            min-elements 1;
            description
                "Supported endpoint properties."
        }
        uses algorithm;
    }
}
case propmap {
   if-feature "propmap";
   container alto-propmap-params {
      uses algorithm;
      description
      "(Filtered) Entity Property Map specific configuration";
   }
}

case cdni {
   if-feature "cdni";
   container alto-cdni-params {
      uses algorithm;
      description
      "CDNi specific configuration";
   }
}

case update {
   if-feature "incr-update";
   container alto-update-params {
      uses algorithm;
      description
      "Incremental Updates specific configuration";
   }
}

description
"ALTO information resources to be defined";

list data-source {
   key "source-id";
   leaf source-id {
      type string;
      description
      "Data source id that can be referenced by information resource creation algorithms.";
   }
   leaf source-type {
      type identityref {
         base source-types;
      }
      mandatory true;
      description
      "Source-type to be defined";
   }
   choice update-policy {
      Z...
mandatory true;
case reactive {
  leaf reactive {
    type boolean;
    mandatory true;
    description
    "Reactive mode";
  }
}
case proactive {
  leaf poll-interval {
    type uint32;
    mandatory true;
    description
    "Polling interval in seconds for proactive mode";
  }
}
description
"Policy to get updates from data sources";
} choice source-params {
case yang-datastore {
  container yang-datastore-source-params {
    leaf source-path {
      type yang:xpath1.0;
      mandatory true;
      description
      "XPath to subscribed YANG datastore node";
    }
    description
    "YANG datastore specific configuration";
  }
}
case prometheus {
  container prometheus-source-params {
    leaf source-uri {
      type inet:uri;
      mandatory true;
      description
      "URI to prometheus agent";
    }
    leaf query-data {
      type string;
      description
      "Query expression";
    }
    description
    "Prometheus specific configuration";
  }
8.2. The ietf-alto-stats Module

This YANG module defines all the configured and operational parameters of the administrated ALTO server instance.

Copyright (c) 2022 IETF Trust and the persons identified as authors of the code. All rights reserved.
Redistribution and use in source and binary forms, with or without modification, is permitted pursuant to, and subject to the license terms contained in, the Revised BSD License set forth in Section 4.c of the IETF Trust’s Legal Provisions Relating to IETF Documents (https://trustee.ietf.org/license-info).

This version of this YANG module is part of RFC XXXX (https://www.rfc-editor.org/info/rfcXXXX); see the RFC itself for full legal notices.

revision "2022-03-07" {
  description
    "Initial Version.";
  reference
    "RFC XXXX: A YANG Data Model for Operations, Administration, and Maintenance of ALTO Protocol.";
}

augment "/alto:alto-server/alto:resource" {
  description
    "Common statistics for each information resource.";
  leaf num-res-upd {
    type yang:counter32;
    config false;
    description
      "The number of version updates since the information resource was created.";
  }
  leaf res-mem-size {
    type yang:counter32;
    config false;
    description
      "Memory size (Bytes) utilized by the information resource.";
  }
  leaf res-enc-size {
    type yang:counter32;
    config false;
    description
      "Size (Bytes) of JSON encoded data of the information resource.";
  }
}

  description
    "Augmented statistics for network maps only.";
}
leaf num-map-pid {
  type yang:counter32;
  config false;
  description
    "Number of PIDs contained by the network map.";
}

augment "/alto:alto-server/alto:resource/alto:resource-params"
  + "/alto:propmap/alto:alto-propmap-params" {
  description
    "Augmented statistics for property maps only.";
  leaf num-map-entry {
    type yang:counter32;
    config false;
    description
      "Number of ALTO entities contained by the property map.";
  }
}

augment "/alto:alto-server/alto:resource/alto:resource-params"
  + "/alto:cdni/alto:alto-cdni-params" {
  description
    "Augmented statistics for CDNi resources only.";
  leaf num-base-obj {
    type yang:counter32;
    config false;
    description
      "Number of base CDNi advertisement objects contained by the
      CDNi resource.";
  }
}

augment "/alto:alto-server/alto:resource/alto:resource-params"
  + "/alto:update/alto:alto-update-params" {
  description
    "Augmented statistics for incremental updates only.";
  leaf num-upd-sess {
    type yang:counter32;
    config false;
    description
      "Number of sessions connected to the incremental update
      service.";
  }
  leaf num-event-total {
    type yang:counter32;
    config false;
    description
    "Number of events reported by incremental updates.";
  }
}

leaf num-map-pid {
  type yang:counter32;
  config false;
  description
    "Number of PIDs contained by the network map.";
}
"Total number of update events sent to all the connected clients."
}
leaf num-event-max {
  type yang:counter32;
  config false;
  description
    "The maximum number of update events sent to the connected clients.";
}
leaf num-event-min {
  type yang:counter32;
  config false;
  description
    "The minimum number of update events sent to the connected clients.";
}
leaf num-event-avg {
  type yang:counter32;
  config false;
  description
    "The average number of update events sent to the connected clients.";
}

9. Security Considerations

TBD.

10. IANA Considerations

This document registers two URIs in the "IETF XML Registry" [RFC3688]. Following the format in RFC 3688, the following registrations are requested.

Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

This document registers two YANG modules in the "YANG Module Names" registry [RFC6020].
Name: ietf-alto
Prefix: alto
Reference: [RFCthis]

Name: ietf-alto-stats
Prefix: alto
Reference: [RFCthis]

[RFC Editor: Please replace RFCthis with the published RFC number for this document.]

11. References

11.1. Normative References


11.2. Informative References


[I-D.ietf-alto-cdni-request-routing-alto]
Appendix A. Example Module for Information Resource Creation Algorithm

The base data model defined by ietf-alto.yang does not include any choice cases for information resource creation algorithms. But developers may augment the ietf-alto.yang data model with definitions for any custom creation algorithms for different information resources. The following example module demonstrates the parameters of a network map creation algorithm that translates an IETF layer 3 unicast topology into a network map.
module example-ietf-alto-alg {
    namespace "urn:example:ietf-alto-alg";
    prefix "alto-alg";

    import ietf-alto {
        prefix "alto";
    }

        case 13-unicast-cluster {
            container 13-unicast-cluster-algorithm {
                leaf 13-unicast-topo {
                    type leafref {
                        path "/alto:alto-server/data-source/source-id";
                    }
                    mandatory true;
                    description "The data source to an IETF layer 3 unicast topology.";
                }
                leaf depth {
                    type uint32;
                    description "The depth of the clustering.";
                }
            }
        }
    }
}

Acknowledgements

The authors thank Qiufang Ma and Qin Wu for their help with drafting the initial version of the YANG modules. Thanks also to Adrian Farrel, Qiao Xiang, Qin Wu, and Qiufang Ma for their reviews and valuable feedback.

Authors’ Addresses

Jingxuan Jensen Zhang
Tongji University
4800 Cao’An Hwy
Shanghai
201804
China
Email: jingxuan.n.zhang@gmail.com
Dhruv Dhody
Huawei Technologies
Divyashree Techno Park, Whitefield
Bangalore 560066
Karnataka
India
Email: dhruv.ietf@gmail.com

Kai Gao
Sichuan University
No.24 South Section 1, Yihuan Road
Chengdu
610000
China
Email: kaigao@scu.edu.cn

Roland Schott
Deutsche Telekom
Heinrich-Hertz-Strasse 3-7
64295 Darmstadt
Germany
Email: Roland.Schott@telekom.de