ALTO WG
Internet-Draft
Intended status: Informational
Expires: January 12, 2023

LM. Contreras
Telefonica

July 11, 2022

Considering ALTO as IETF Network Exposure Function
draft-contreras-alto-ietf-nef-01

Abstract

This document proposes ALTO as the means for exposure of underlay network capabilities for multiple overlays on top of the network.

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 January 12, 2023.

Table of Contents

1. Introduction

Networks are turning on consumable objects by external applications and services. In order to facilitate that, it is necessary to expose the capabilities offered by the networks in such a way that the applications and services can produce informed decisions that assist in the improvement of the service delivery.

Thus it is convenient to define mechanisms for capabilities exposure that could provide required information for IETF networks. ALTO [RFC7285] can play such a role. This memo describes existing and foreseen capabilities that can be exposed by leveraging on ALTO.

2. Exposing network capabilities for enhancing service delivery

More and more, services and applications rely on information retrieved from the network in order to make decisions positively affecting the service delivery, by adapting the applications to the reality observed through the retrieved information. This information is typically offered by specific components in the network with the

mission of aggregating, processing and securely exposing such information.

Several initiatives are being developed in order to facilitate such exposure of capabilities and information at different network levels. For example, 3GPP defines the Network Exposure function (NEF) [TS29.522] as a secure, scalable and simplified exposing tool for capabilities (as well as events) supported by the 5G Core (5GC) network. Main capabilities of NEF are the following:

- Securely expose 3GPP Network Functions (NFs) capabilities to Application Functions (AF).
- Secure provision of information to 5GC, including authentication and authorization to AF.
- Rate limit AF access to 5GC functions and information, including charging functions.
- Translation of internal-external information, e.g. identities.

This is done though a number of APIs defined in [TS29.522]. A specific NEF instance may support only a subset of the APIs specified for capability exposure.

Further examples are present in other network concerns. Thus, in ETSI Multi-Access Edge Computing (MEC) group a number of APIs allow the retrieval of specific network information at the edge (e.g., location API [MEC-013]), or the O-RAN Alliance which is working on exposing information to applications running on top of the non-real time Radio Informacion Controller (RIC) [O-RAN].

The purpose of this document is to consider ALTO as the means for exposure of underlay network capabilities to multiple overlays on top of the network. In other words, serve as "ground truth" from the network provider perspective to the applications consuming network capabilities in the scope of IETF.

3. ALTO versus network controller conceptualization

A relevant question that could arise is about the difference on purpose between ALTO and a network controller in the network.

Primarily, the final purpose of these components is quite different. In this respect, a network controller (i.e., SDN controller [RFC7149]) can be seen as the element devoted to orchestration, control and management of the network assets, that is, the component in charge of administering network objects. Typically, a network

controller leverages on another IETF functional component used for network control, such as the Path Computation Element (PCE) [RFC4655], which is used to compute paths for forwarding purposes based on network constraints. In contrast to these two elements, ALTO acts as a "one-stop-shop" for retrieving (and correlating) network related information, potentially leveraging on the capabilities of the othe rtow (i.e., SDN controller and/or PCE).

Moreover, ALTO has been included as part of some architectural frameworks, such ABNO [RFC7491], with the mission of allowing joint network and application-layer optimization precisely by exposing to applications an abstract network topology containing only information relevant to such application. In this manner the application can take an informed decision and request specific control actions in the network.

4. Modes of usage

This section presents different modes of usage of ALTO network exposure capabilities to improve network operations. Some of these usages can be implemented nowadays based on existing specifications, while a set of other use cases is considered as prospective since more specification work is yet needed in IETF.

4.1. Existing use cases

This subsection presents a number of use cases already described that can leverage on ALTO as IETF Network Exposure Function.

4.1.1. Network topology and reachability

The basic ALTO capabilities [RFC7285] provide network maps associated with costs in a manner that for any pair of source and destination can be retrieved information about topology and reachability. This can be considered as the fundamental or baseline information on top of which the other modes of usage are built on.

4.1.2. Network performance metrics per path

Extensions defined in [I-D.ietf-alto-performance-metrics] permit the reporting of standard-based performance metrics associated to the paths generated in the network map. With that view, applications consuming ALTO (i.e., ALTO clients) can determine the performance expectation for the possible paths between an origin and a destination. Thus, not only puere cost but also performance can be considered as an element for decision.

4.1.3. Segmented paths and associated characteristics

Original ALTO concentrates on end-to-end paths. However it may result of interest to get knowledge of specific parts of the end-to-end paths that could produce problems such as e.g. congestion. Then, having means of segmenting the end-to-end paths becomes useful. [I-D.ietf-alto-path-vector] allows for that defining a new abstraction called Abstract Network Element (ANE) to represent components constituting an entire end-to-end network path as a vector of ANEs.

4.1.4. In-time view of dynamic IP addressing allocation

Some architectures allow for dynamic allocation of IP address subnets across the network. An example of that is the Control and User Plane Separation (CUPS) architecture for Broadband Network Gateways (BNGs) [I-D.wadhwa-rtgwg-bng-cups], [TR-459]. In that architecture, the control place of the BNG has the possibility of dynamically assigning IP address subnets to different elements distributed in the network, acting as user plane functions of the BNG. This dynamic allocation implies that certain IP prefixes could be allocated in different parts of the network along the time. By means of ALTO and its network map is it possible to obtain an up-to-date view of the topological location of each subnet in runtime, facilitating the optimization of some services (e.g. media distribution) in an automated manner.

4.2. Prospective use cases

This subsection presents a number of use cases that could be enabled by ALTO as IETF Network Exposure Function.

4.2.1. Determination of optimal compute facility taking into account network information

ALTO can be used as a component to provide insights on the reachability of suitable compute facilities. An initial case has been documented in [I-D.contreras-alto-service-edge]. The rationale for this case is that ALTO receives information of connected compute capabilities in terms of e.g. CPU, memory and storage. This information can be put together with the network map, in a way that the cost of reaching those capabilities can be easily determined.

Note that if further information apart of cost is included in the map (e.g., performance metrics) then the resulting information provided to applications becomes enriched.

4.2.2. Information related to Service Functions and Service Function chains

ALTO can provide information relative to the paths characteristics associated with a single Service Function or with a number of chained Service Functions. This can be useful at the definition phase of a network service, either considering specific instances of the constituent Service Functions, or as a mean of identifying the more appropriate Service Functions to compose a service.

[I-D.lcsr-alto-service-functions] proposes different situations of interest and explores augmentations in ALTO to support the retrieval of information associated to Service Functions. Internal IETF solutions as the ones for Service Function Chaining or SRv6 programmability can benefit of this insight, but also other solutions like ETSI NFV, 3GPP, O-RAN or any other requiring efficient decisions in relation with chains of Service Functions can be benefitted for their own automation, management and control processes.

4.2.3. Visibility of underlying network information in overlay networks

Different overlay networks run today leveraging the connectivity provided by the basic underlying transport network. Since specific situations on the transport network can result in relevance for the service being provided by the overlays, it is crucial to facilitate the observation of such situations from the underlay to the overlay.

4.2.3.1. Cellular case

Mobile networks leverage transport networks to connect mobile access nodes with core management and control entities (e.g., for mobility management, policing, etc), running in an overlay mode through tunneling (i.e., the GTP protocol). [I-D.li-alto-cellular-use-cases] presents the benefits of exposing network information for applications running on access devices of a cellular network.

4.2.3.2. Media distribution case

Media delivery systems, as traditional CDNs, deliver content to end-users in an over-the-top fashion. The key aspect for an efficient and optimal delivery of the content is to select the proper delivery point for whatever end-user requesting it is to have a clear view of the network topology (including the associated costs or any other information that could enrich the decision, such as performance metrics). In this respect, the information exposed by ALTO in reference to the requesting end-user can be consumed by CDN control elements for improving the decision on what delivery point to select [RFC7971].

Further than that, additional scenarios can benefit from ALTO network information exposure capabilities. For instance, in scenarios of interconnection of CDNs, such as the one described in [I-D.ryan-cdni-capacity-insights-extensions] for advertising capacity associated with the CDN internal to an operator, could leverage on ALTO capabilities for that purpose (with the necessary augmentations).

5. ALTO as IETF Network Exposure Function

From its inception, ALTO was defined as a way of informing applications about network-related aspects for improving the overall service.

The applications under scope can be either internal or external to the operator of the network. The implications can differ in the level of aggregating, abstracting and securely exposing the information, but the purpose keeps being the same.

Figure 1 illustrates the role of ALTO as IETF Network Exposure Function.

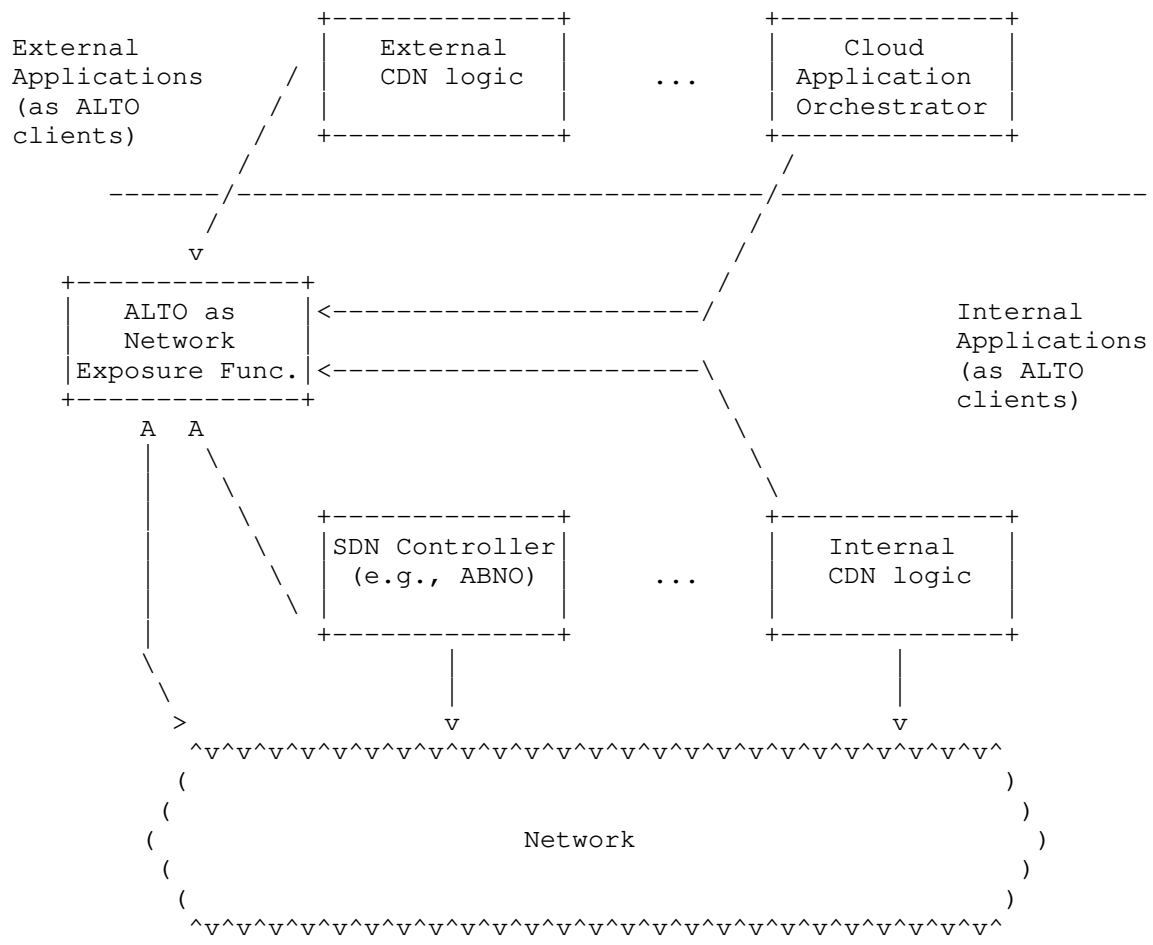


Figure 1: ALTO as IETF Network Exposure Function

Each particular ALTO instance in a certain network could support only a subset of the capabilities discussed in the use cases described before. In this respect, ways of advertising supported capabilities should be defined.

6. TODO for next versions of this document

This version is a work-in-progress. Next versions of the document will address somo further aspects such as:

- Identification of further network capabilities of interest for being exposed by ALTO.

- Complete security aspects when exposing information to internal and external applications.

7. Security Considerations

ALTO security considerations as reflected in [RFC7285] apply to this document.

Apart from that, the following aspects should be taken into consideration:

- Authentication between ALTO and any external entitiy consuming ALTO, to prevent malicious behaviors.

- Privacy of the information shared between components, especially when those components pertain to different administrative domain (e.g., an external CDN retieving network information from a network of a different administrative domain).

- Secure transport of the information in the communication with ALTO Server (e.g., TLS, etc).

8. IANA Considerations

This draft does not include any IANA considerations

9. References

[I-D.contreras-alto-service-edge]
Contreras, L. M., Lachos, D. A., Rothenberg, C. E., and S. Randriamasy, "Use of ALTO for Determining Service Edge", draft-contreras-alto-service-edge-05 (work in progress), July 2022.

[I-D.ietf-alto-path-vector]
Gao, K., Lee, Y., Randriamasy, S., Yang, Y. R., and J. J. Zhang, "An ALTO Extension: Path Vector", draft-ietf-alto-path-vector-25 (work in progress), March 2022.

[I-D.ietf-alto-performance-metrics]
Wu, Q., Yang, Y. R., Lee, Y., Dhody, D., Randriamasy, S., and L. M. C. Murillo, "ALTO Performance Cost Metrics", draft-ietf-alto-performance-metrics-28 (work in progress), March 2022.

[I-D.lcsr-alto-service-functions]
Contreras, L. M. and S. Randriamasy, "ALTO extensions for handling Service Functions", draft-lcsr-alto-service-functions-00 (work in progress), July 2022.

[I-D.li-alto-cellular-use-cases]
Gang, L., Randriamasy, S., and C. Xiong, "ALTO Uses Cases for Cellular Networks", draft-li-alto-cellular-use-cases-00 (work in progress), July 2021.

[I-D.ryan-cdni-capacity-insights-extensions]
Ryan, A., Rosenblum, B., and N. B. Sopher, "CDNI Capacity Capability Advertisment Extensions", draft-ryan-cdni-capacity-insights-extensions-02 (work in progress), March 2022.

[I-D.wadhwa-rtgwg-bng-cups]
Wadhwa, S., Shinde, R., Newton, J., Hoffman, R., Muley, P., and S. Pani, "Architecture for Control and User Plane Separation on BNG", draft-wadhwa-rtgwg-bng-cups-03 (work in progress), March 2019.

[MEC-013] "GS MEC 013 Location API V2.1.1", ETSI GS MEC 013 V2.1.1 , September 2019.

[O-RAN] "Non-RT RIC Architecture", O-RAN.WG2.Non-RT-RIC-ARCH-TS-v01.00.02 , July 2021.

[RFC4655] Farrel, A., Vasseur, J., and J. Ash, "A Path Computation Element (PCE)-Based Architecture", RFC 4655, DOI 10.17487/RFC4655, August 2006, <https://www.rfc-editor.org/info/rfc4655>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC7149] Boucadair, M. and C. Jacquenet, "Software-Defined Networking: A Perspective from within a Service Provider Environment", RFC 7149, DOI 10.17487/RFC7149, March 2014, <https://www.rfc-editor.org/info/rfc7149>.

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <https://www.rfc-editor.org/info/rfc7285>.

[RFC7491] King, D. and A. Farrel, "A PCE-Based Architecture for Application-Based Network Operations", RFC 7491, DOI 10.17487/RFC7491, March 2015, <https://www.rfc-editor.org/info/rfc7491>.

[RFC7971] Stiemerling, M., Kiesel, S., Scharf, M., Seidel, H., and S. Previdi, "Application-Layer Traffic Optimization (ALTO) Deployment Considerations", RFC 7971, DOI 10.17487/RFC7971, October 2016, <https://www.rfc-editor.org/info/rfc7971>.

[TR-459] "Control and User Plane Separation for a disaggrgated BNG", Broadband Forum TR-459 , June 2020.

[TS29.522] "TS 29.522 Network Exposure Function Northbound APIs V16.9.0.", 3GPP TS 29.522 V16.9.0 , September 2021.

Acknowledgments

...

Author's Address

Luis M. Contreras
Telefonica
Ronda de la Comunicacion, s/n
Sur-3 building, 3rd floor
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com
URI: http://lmcontreras.com/

ALTO WG
Internet-Draft
Intended status: Informational
Expires: January 12, 2023

LM. Contreras
Telefonica
D. Lachos
BENOCS
C. Rothenberg
Unicamp
S. Randriamasy
Nokia Bell Labs

July 11, 2022

Use of ALTO for Determining Service Edge
draft-contreras-alto-service-edge-05

Abstract

Service providers are starting to deploy and interconnect computing capabilities across the network for hosting network functions and applications. In distributed computing environments, both computing and topological information are necessary in order to determine the more convenient infrastructure where to deploy such a service or application. This document proposes an initial approach towards the use of ALTO to provide such information and assist the selection of appropriate deployment locations for services and applications.

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 January 12, 2023.

Copyright Notice

Table of Contents

1. Introduction

The advent of virtualization is enabling the operators with a dynamic instantiation of network functions and applications by using different techniques on top of commoditized computation infrastructures, permitting a flexible and on-demand deployment of services, aligned with the actual needs observed as demanded by the customers.

Operators are starting to deploy distributed computing environments in different parts of the network with the objective of addressing the different service needs in terms of latency, bandwidth, processing capabilities, etc. This is translated in the emergence of a number of data centers of different sizes (e.g., large, medium, small) characterized by distinct dimension of CPUs, memory and storage capabilities, as well as bandwidth capacity for forwarding the traffic generated in and out the corresponding data center.

The probable future situation, with the generalization and proliferation of the edge computing approach, will increase the

potential footprint where a function or application can be deployed. These different dimensioning rules result in a different unitary cost per CPU, memory, and storage in each computing environment because of the scale.

All the available distributed computing capabilities can complicate the decision of what infrastructure use for instantiating a given function or application. Such a decision influences not only the resources that are consumed in a given computing environment, but also the network capacity of the path that connects such environment with the rest of the network from traffic source to destination.

It is then essential for a network operator to have mechanisms assisting on the decision by considering a number of constraints related to the function or application to be deployed understanding how a given decision on the computing environment for the service edge affects to the transport network substrate. This would allow to integrate network capabilities in the function placement decision and further optimize performance of the deployed application.

This document proposes the usage of ALTO [RFC7285] for assisting with such a decision.

2. Computing needs

A given network function or application typically shows certain requirements in terms of processing capabilities (i.e., CPU), as well as volatile memory (i.e., RAM) and storage capacity.

Cloud computing providers, such as Amazon Web Services or Microsoft Azure, typically structure their offerings of computing capabilities by bundling CPU, RAM and storage units as quotas, instances or flavors that can be consumed in an ephemeral or temporal fashion, during the actual lifetime of the required function or application.

This same approach is being taken nowadays for characterizing bundles of resources on the so-called Network Function Virtualization Infrastructure (NFVI) Points of Presence (PoPs) being deployed by the telco operators. Specifically, the Common Network Function Virtualisation Infrastructure Telecom Taskforce (CNTT) [CNTT], [GSMA], jointly hosted by GSMA and the Linux Foundation, is intending to harmonize the definition of above-mentioned computing capability instances or flavors for abstracting capabilities of the underlying NFVI facilitating a more efficient utilization of the infrastructure and simplifying the integration and certification of functions, where certification means the assessment of the expected behavior for a given function according to the leverl of resources determined by a given flavor.

Focusing on the CNTT ongoing work, the flavors or instances are characterized according to:

- Type of instance (T): the types of instances are characterized as B (Basic), or N (Network Intensive). The latter can come with extensions for network acceleration for offloading network intensive operations to hardware.

- Interface Option (I): it refers to the associated bandwidth of the network interface.

- Compute flavor (F): it refers to a given predefined combination of resources in terms of virtual CPU, RAM, disk, and bandwidth for the management interface.

- Optional storage extension (S): allows to request additional storage capacity.

- Optional hardware acceleration characteristics (A): to request specific acceleration capabilities for improving the performance of the infrastructure.

The naming convention of an instance is thus encoded as T.I.F.S.A.

3. Usage of ALTO for determining where to deploy a function or application

ALTO can assist the deployment of a service or application on a specific flavor or instance of the computing substrate by taking into consideration network cost metrics.

A generic and primary approach is to take into account metrics related to the computing environment, such as availability of resources, unitary cost of those resources, etc.

Nevertheless, the function or application to be deployed on top of a given flavor is interconnected outside the computing environment where it is deployed, also requiring to guarantee transport network requirements to ensure the application performance, such as bandwidth, latency, etc.

The objective then is to leverage on ALTO to provide information about the more convenient execution environments to deploy virtualized network functions or applications, allowing the operator to get a coordinated service edge and transport network recommendation.

3.1. Integrating compute information in ALTO

CNTT proposes the existence of a catalogue of compute infrastructure profiles collecting the computing capability instances available to be consumed. Such kind of catalogue could be communicated to ALTO or even incorporated to it.

ALTO server queries are required to support T.I.F.S.A encoding in order to retrieve proper maps from ALTO. Additionally, filtered queries for particular characteristics of a flavor could also be supported.

3.2. Association of compute capabilities to network topology

It is required to associate the location of the available instances with topological information to allow ALTO construct the overall map. The expectation is to manage the network and cloud capabilities by the same entity, handling both network and compute abstractions jointly, producing an integrated map. While this can be straightforward when an ISP own both the network and the cloud infrastructure, it could require multiple administrative domains to interwork for composing the integrated map. Details on potential scenarios will be provided in future versions of the document.

At this stage four potential solutions could be considered:

- To leverage on (and possibly extend) [I-D.ietf-teas-sf-aware-topo-model] for disseminating topology information together with notion of function location (that would require to be adapted to the existence of available compute capabilities). A recent effort in this direction can be found in [I-D.llc-teas-dc-aware-topo-model].

- To extend BGP-LS [RFC7752], which is already considered as mechanism for feeding topology information in ALTO, in order to also advertise computing capabilities as well.

- To combine information from the infrastructure profiles catalogue with topological information by leveraging on the IP prefixes allocated to the gateway providing connectivity to the NFVI PoP.

- To integrate with Cloud Infrastructure Managers that could expose cloud infrastructure capabilities as in [CNTT], [GSMA].

The viability of these options will be explored in future versions of this document.

3.3. ALTO architecture for determining serve edge

The following logical architecture defines the usage of ALTO for determining service edges.

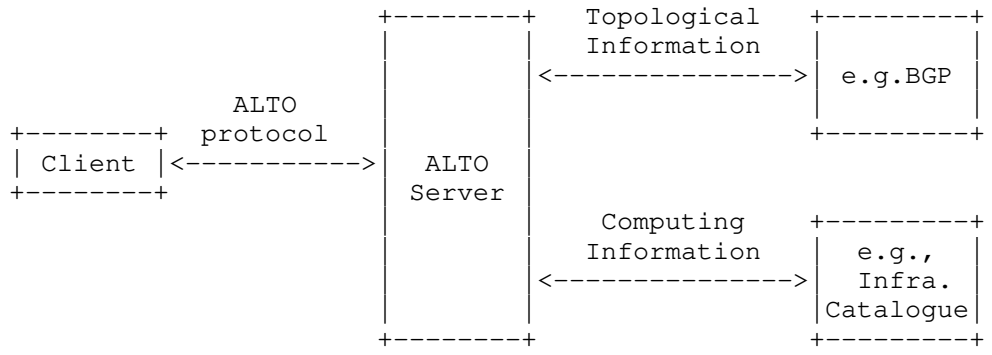


Figure 1: Service Edge Information Exchange

In order to select the optimal edge server from both the network perspective (e.g., the one showing better path cost) and cloud capabilities (e.g. in terms of processing capabilities, available RAM, storage capacity, etc), it is needed to see an edge server as both an IP entity (as in [RFC7285]) and an ANE entity (as in [I-D.ietf-alto-path-vector]).

Currently there is no way (neither in [I-D.ietf-alto-path-vector] nor [DRAFT-PM]) to see the same edge server as an entity in both domains. One possible way, for instance, can be to introduce entity properties that list other entity domains where an edge server is identified. Thus, if an edge server is identified by:

- in the IPV4 domain, with an IP address, e.g. ipv4:1.2.3.4
- in the ANE domain, with an identifier, e.g. ane:DC10-HOST1

With this, a potential solution can be:

```
--- in the IP entity domain
ipv4:1.2.3.4
Properties:
   pid : DC10
   entity domain mapping : [<<ane>>]

--- in the ANE domain
ane:DC10-HOST1
Properties:
   entity domain mapping :  : [<<ipv4>>]
   Name: DC10-HOST1
   network-address : ipv4:1.2.3.4
```

Further elaboration will be provdided in next versions of this document.

4. Definition of flavors in ALTO property map

The ALTO unified property extension [DRAFT-PM] generalizes the concept of endpoint properties to domains of other entities through property maps. In the context of the CNTT domain, an ALTO property map could be used to expose T.I.F.S.A information of potential candidate flavors, i.e., potential NFVI PoPs where an application or service can be deployed.

Figure 2 below shows an illustrative example of an ALTO property map with property values grouped by flavor name.

flavor -name	type (T)	inter face (I)	f-c pu (F)	f-ra m (F)	f-di sk (F)	f-b w (F)	S	A
small- 1	basic	1 Gbps	1	512 MB	1 GB	1 G bps		
small- 2	network- intensive	9 Gbps	1	512 MB	1 GB	1 G bps		
medium -1	network- intensive	25 Gbps	2	4 GB	40 GB	1 G bps		
large- 1	compute- intensive	50 Gbps	4	8 GB	80 GB	1 G bps		
large- 2	compute- intensive	100 Gbps	8	16 GB	160 GB	1 G bps		

Figure 2: ALTO Property Map

The following example uses the filtered property map resource to request properties "type", "cpu", "ram", and "disk" on several flavor names defined in the previous property map.

```
POST /propmap/lookup/ane-flavor-name HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: 155
Content-Type: application/alto-propmapparams+json
{
  "entities" : ["small-1",
                "small-2",
                "medium-1",
                "large-2"],
  "properties" : [ "type", "cpu", "ram", "disk"]
}

HTTP/1.1 200 OK
Content-Length: 295
Content-Type: application/alto-propmap+json
{
  "meta" : {
  },
  "property-map": {
    "small-1":
      {"type" : "basic", "cpu" : 1,
        "ram" : "512MB", "disk" : 1GB},
    "small-2":
      {"type" : "network-intensive", "cpu" : 1,
        "ram" : "512MB", "disk" : 1GB},
    "medium-1":
      {"type" : "compute-intensive", "cpu" : 4,
        "ram" : "8GB", "disk" : 80GB},
    "large-2":
      {"type" : "compue-intensive", "cpu" : 8,
        "ram" : "16GB", "disk" : 160GB},
  }
}
```

Figure 3: Filtered Property Map query example

5. IANA Considerations

This document includes no request to IANA.

6. Security Considerations

TBD.

7. Conclusions

Telco networks will increasingly contain a number of interconnected data centers, of different size and characteristics, allowing flexibility in the dynamic deployment of functions and applications for advance services. The overall objective of this document is to begin a discussion in the ALTO WG regarding the suitability of the ALTO protocol for determining where to deploy a function or application in distributed computing environments. The result of such discussions will be reflected in future versions of this draft.

8. References

8.1. Normative References

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <https://www.rfc-editor.org/info/rfc7285>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

[RFC8189] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost Application-Layer Traffic Optimization (ALTO)", RFC 8189, DOI 10.17487/RFC8189, October 2017, <https://www.rfc-editor.org/info/rfc8189>.

8.2. Informative References

[CNTT] "Cloud iNfrastructure Telco Taskforce Reference Model", <https://github.com/cntt-n/CNTT/wiki>.

[DRAFT-PM] Roome, W., Randriamasy, S., Yang, Y., Zhang, J., and K. Gao, "Unified Properties for the ALTO Protocol", draft-ietf-alto-unified-props-new-24 (work in progress), February 2022.

[GSMA] "Cloud Infrastructure Reference Model, Version 2.0", October 2021, <https://www.gsma.com/newsroom/wp-content/uploads//NG.126-v2.0-1.pdf>.

[I-D.ietf-alto-path-vector]
Gao, K., Lee, Y., Randriamasy, S., Yang, Y. R., and J. J. Zhang, "An ALTO Extension: Path Vector", draft-ietf-alto-path-vector-25 (work in progress), March 2022.

[I-D.ietf-teas-sf-aware-topo-model]
Bryskin, I., Liu, X., Lee, Y., Guichard, J., Contreras, L., Ceccarelli, D., Tantsura, J., and D. Shytyi, "SF Aware TE Topology YANG Model", draft-ietf-teas-sf-aware-topo-model-09 (work in progress), February 2022.

[I-D.llc-teas-dc-aware-topo-model]
Lee, Y., Liu, X., and L. Contreras, "DC aware TE topology model", draft-llc-teas-dc-aware-topo-model-01 (work in progress), July 2021.

Authors' Addresses

Luis M. Contreras
Telefonica
Ronda de la Comunicacion, s/n
Madrid 28050
Spain

Email: luismiguel.contrerasmurillo@telefonica.com
URI: http://lmcontreras.com/

Danny Alex Lachos Perez
BENOCS GmbH
Reuchlinstrasse 10
Berlin 10553
Germany

Email: dlachos@benocs.com

Christian Esteve Rothenberg
University of Campinas
Av. Albert Einstein 400
Campinas, Sao Paulo 13083-970
Brazil

Email: chesteve@dca.fee.unicamp.br
URI: https://intrig.dca.fee.unicamp.br/christian/

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
Nozay 91460
France

Email: sabine.randriamasy@nokia-bell-labs.com

ALTO
Internet-Draft
Intended status: Informational
Expires: 15 April 2024

L. M. Contreras
Telefonica
S. Randriamasy
Nokia Bell Labs
J. Ros-Giralt
Qualcomm Europe, Inc.
D. Lachos
Benocs
C. Rothenberg
Unicamp

13 October 2023

Use of ALTO for Determining Service Edge
draft-contreras-alto-service-edge-10

Abstract

Service providers are starting to deploy computing capabilities across the network for hosting applications such as AR/VR, vehicle networks, IoT, and AI training, among others. In these distributed computing environments, knowledge about computing and communication resources is necessary to determine the proper deployment location of each application. This document proposes an initial approach towards the use of ALTO to expose such information to the applications and assist the selection of their deployment locations.

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-contreras-alto-service-edge/#go.draft-contreras-alto-service-edge.html. Status information for this document may be found at https://datatracker.ietf.org/doc/draft-contreras-alto-service-edge/.

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/. Subscribe at https://www.ietf.org/mailman/listinfo/alto/.

Source for this draft and an issue tracker can be found at https://github.com/giralt/draft-contreras-alto-service-edge.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

With the advent of network virtualization, operators can make use of dynamic instantiation of network functions and applications by using different techniques on top of commoditized computation infrastructures, permitting a flexible and on-demand deployment of services, aligned with the actual needs as demanded by the users.

Operators are starting to deploy distributed computing environments in different parts of the network with the objective of addressing different service needs including latency, bandwidth, processing capabilities, storage, etc. This is translated in the emergence of a number of data centers (both in the cloud and at the edge) of different sizes (e.g., large, medium, small) characterized by distinct dimension of CPUs, memory, and storage capabilities, as well as bandwidth capacity for forwarding the traffic generated in and out of the corresponding data center.

The proliferation of the edge computing paradigm further increases the potential footprint and heterogeneity of the environments where a function or application can be deployed, resulting in different unitary cost per CPU, memory, and storage. This increases the complexity of deciding the location where a given function or application should be best deployed, as this decision should be influenced not only by the available resources in a given computing environment, but also by the network capacity of the path connecting the traffic source with the destination.

It is then essential for a network operator to have mechanisms assisting this decision by considering a number of constraints related to the function or application to be deployed, understanding how a given decision on the computing environment for the service edge affects the transport network substrate. This would enable the integration of network capabilities in the function placement decision and further optimize performance of the deployed application.

This document proposes the use of ALTO [RFC7285] for assisting such a decision.

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. Computing Needs

The compute needs for an application or service function are typically set in terms of certain requirements in terms of processing capabilities (i.e., CPU), as well as volatile memory (i.e., RAM) and storage capacity. There are two ways of considering those needs: either as individual values for each of the resources, or as a pre-defined bundle of them in the form of instance or compute flavor.

3.1. Working with compute resource values

Compute resource values can be obtained from cloud manager systems able to provide information about compute resources in a given compute node. These cloud manager systems typically provide information about total resources in the compute node and either the used or the allocatable resources. This information can be leveraged for taking application or service function placement decisions from the compute capability perspective. Each cloud management system has their own schema of information to be provided. In general terms, information about CPU, memory and storage can be provided, together with the identifiers of the compute node and some other system information. Examples of these cloud management systems are Kubernetes, OpenStack and OpenNebula. The following table summarizes some of the obtainable information from these cloud management systems.

Cloud Management System	Basic compute information	Additional information (example)
Kubernetes	CPU, memory (total and allocatable)	Node name, machine ID, operating system, etc
OpenStack	CPU, memory, storage (total and used)	Compute node ID, node name, hypervisor type, etc
OpenNebula	CPU, memory, storage (max and used)	Compute node name, cluster ID, hypervisor type, etc

Figure 1: Compute resource information from well known cloud managers

3.2. Working with compute flavors

Cloud computing providers such as Amazon Web Services Microsoft Azure, or Google Cloud Platform, typically structure their offerings of computing capabilities by bundling CPU, RAM and storage units as quotas, instances, and flavors that can be consumed in an ephemeral fashion, during the actual lifetime of the required function or application. In the case of Amazon Web Services such grouping of compute resources is denominated instance type, being the same concept named as virtual machine size in Microsoft Azure and machine type in Google Cloud Platform.

This same approach is being taken for characterizing bundles of resources on the so-called Network Function Virtualization Infrastructure Points of Presence (NFVI-PoPs) being deployed by the telco operators. For instance, the Common Network Function Virtualization Infrastructure Telecom Taskforce (CNTT) [CNTT], jointly hosted by GSMA [GSMA] and the Linux Foundation, intends to harmonize the definition of instances and flavors for abstracting capabilities of the underlying NFVI, facilitating a more efficient utilization of the infrastructure and simplifying the integration and certification of functions. (Here certification means the assessment of the expected behavior for a given function according to the level of resources determined by a given flavor.) An evolution of this initiative is Anuket [Anuket], which works to consolidate different architectures for well-known tools such as OpenStack and Kubernetes.

Taking CNTT as an example, the flavors or instances can be characterized according to:

* _Type of instance (T):_ Used to specify the type of instances, which are characterized as B (Basic), or N (Network Intensive). The latter includes network acceleration extensions for offloading network intensive operations to hardware.

* _Interface Option (I):_ Used to specify the associated bandwidth of the network interface.

* _Compute flavor (F):_ Used to specify a given predefined combination of resources in terms of virtual CPU, RAM, disk, and bandwidth for the management interface.

* _Optional storage extension (S):_ Used to specify additional storage capacity.

* _Optional hardware acceleration characteristics (A):_ Used to specify acceleration capabilities for improving the performance of the application.

The naming convention of an instance is thus encoded as TIFSA.

3.3. ALTO to support abstracted compute information

Examples in sections 3.1 in particular Figure 1 stress the need to abstract these values for the sake of harmonization and ALTO could provide the information services to do so. Abstraction is also needed to (i) aggregate values for simplicity or scalability and (ii) support potential confidentiality needs of data center management entity. To specify the ALTO metrics relevant to compute capabilities, an exercise similar to the RFC-9439 to be [I-D.ietf-alto-performance-metrics] would be useful. The initial metrics could be taken from different standardization bodies or cloud providers or IETF working groups. Besides, metrics reflecting energy consumption of application deployment footprint also need to be considered, given the expected massive usage of ML/AI and the current context urging to optimize energy consumption.

4. Usage of ALTO for Service Placement

ALTO can assist the deployment of a service on a specific flavor or instance of the computing substrate by taking into consideration network cost metrics. A generic and primary approach is to take into account metrics related to the computing environment, such as availability of resources, unitary cost of those resources, etc. Nevertheless, the function or application to be deployed on top of a given flavor must also be interconnected outside the computing environment where it is deployed, therefore requiring the necessary network resources to satisfy application performance requirements

such as bandwidth or latency.

The objective then is to leverage ALTO to provide information about the more convenient execution environments to deploy virtualized network functions or applications, allowing the operator to get a coordinated service edge and transport network recommendation.

4.1. Integrating Compute Information in ALTO

CNTT proposes the existence of a catalogue of compute infrastructure profiles collecting the computing capability instances available to be consumed. Such a catalogue could be communicated to ALTO or even incorporated to it.

ALTO server queries could support TIFSA encoding in order to retrieve proper maps from ALTO. Additionally, filtered queries for particular characteristics of a flavor could also be supported.

4.2. Association of Compute Capabilities to Network Topology

It is required to associate the location of the available instances with topological information to allow ALTO construct the overall map. The expectation is that the management of the network and cloud capabilities will be performed by the same entity, producing an integrated map to handle both network and compute abstractions jointly. While this can be straightforward when an ISP owns both the network and the cloud infrastructure, it can in general require collaboration between multiple administrative domains. Details on potential scenarios will be provided in future versions of this document.

At this stage, four potential solutions could be considered:

* To leverage (and possibly extend) [I-D.ietf-teas-sf-aware-topo-model] for disseminating topology information together with the notion of function location (that would require to be adapted to the existence of available compute capabilities). A recent effort in this direction can be found in [I-D.llc-teas-dc-aware-topo-model].

* To extend BGP-LS [RFC7752], which is already considered as a mechanism for feeding topology information in ALTO, in order to also advertise computing capabilities as well.

* To combine information from the infrastructure profiles catalogue with topological information by leveraging the IP prefixes allocated to the gateway providing connectivity to the NFVI PoP.

* To integrate with Cloud Infrastructure Managers that could expose cloud infrastructure capabilities as in [CNTT] and [GSMA].

The viability of these options will be explored in future versions of this document.

4.3. ALTO Architecture for Determining Serve Edge

The following logical architecture defines the usage of ALTO for determining service edges.

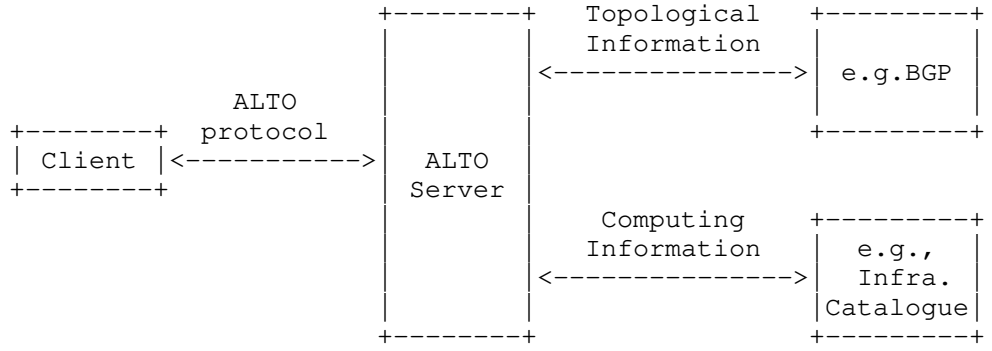


Figure 2: Service Edge Information Exchange.

In order to select the optimal edge server from both the network (e.g., the path with lower latency and/or higher bandwidth) and the cloud perspectives (e.g., number of CPUs/GPUs, available RAM and storage capacity), there is a need to see the edge server as both an IP entity (as in [RFC7285]) and an Abstract Network Element (ANE) entity (as in [RFC9275]).

Currently there is no mechanism (neither in [RFC9275] nor [RFC9240]) to see the same edge server as an entity in both domains. The design of ALTO, however, allows extensions that could be used to identify that an entity can be defined in several domains. These different domains and their related properties can be specified in extended ALTO property maps, as proposed in the next sections.

5. ALTO Design Considerations for Determining Service Edge

This section is in progress and gathers the ALTO features that are needed to support the exposure of both networking capabilities and compute capabilities in ALTO Maps.

In particular, ALTO Entity Property Maps defined in [RFC9240] can be extended. [RFC9240] generalizes the concept of endpoint properties to domains of other entities through property maps. Entities can be defined in a wider set of entity domains such as IPv4, IPv6, PID, AS, ISO3166-1 country codes or ANE. In addition, RFC 9240 specifies how properties can be defined on entities of each of these domains.

5.1. Example of Entity Definition in Different Domains

As there can be applications that do not necessarily need both compute and networking information, it is fine to keep the entity domains separate, each with their own native properties. However, some applications need information on both topics, and a scalable and flexible solution consists in defining an ALTO property type, that:

* Indicates that an entity can be defined in several domains;

* Specifies, for an entity, the other domains where this entity is defined.

For instance, one possible approach is to introduce entity properties that list other entity domains where an edge server is identified. This property type should be usable in all entity domains types. The following provides an example where the property "entity domain mapping" lists the other domains in which an entity is defined.

Suppose an edge server is identified as follows:

* In the IPV4 domain, with an IP address, e.g., ipv4:1.2.3.4;

* In the ANE domain, with an identifier, e.g., ane:DC10-HOST1.

To get information on this edge server as an entity in the "ipv4" entity domain, an ALTO client can query the properties "pid" and "entity-domain-mappings" and obtain a response as follows:

```
POST /propmap/lookup/dc-ip HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: TBC
Content-Type: application/alto-propmapparams+json
{
"entities" : ["ipv4:1.2.3.4"],
"properties" : [ "pid", "entity-domain-mappings"]
}


HTTP/1.1 200 OK
Content-Length: TBC
Content-Type: application/alto-propmap+json
{
"meta" : {},

"property-map":  {
    "ipv4:1.2.3.4" :
      {"pid" : DC10,
        "entity-domain-mappings" : ["ane"]}
    }
}
```

To get information on this edge server as an entity in the "ane" entity domain, an ALTO client can query the properties "entity-domain-mappings" and "network-address" and obtain a response as follows:

```
POST /propmap/lookup/dc-ane HTTP/1.1
Host: alto.example.com
Accept: application/alto-propmap+json,application/alto-error+json
Content-Length: TBC
Content-Type: application/alto-propmapparams+json
{
"entities" : ["ane:DC10-HOST1"],
"properties" : [ "entity-domain-mappings", "network-address"]
}

HTTP/1.1 200 OK
Content-Length: TBC
Content-Type: application/alto-propmap+json
{
"meta" : {},

"property-map":  {
  "ane:DC10-HOST1"
      {"entity domain mappings :  ["ipv4"]",
        "network-address" :  ipv4:1.2.3.4}
      }
}
```

Thus, if the ALTO Client sees the edge server as an entity with a network address, it knows that it can see the server as an ANE on which it can query relevant properties.

Further elaboration will be provided in future versions of this document.

5.2. Definition of Flavors in ALTO Property Map

The ALTO Entity Property Maps [RFC9240] generalize the concept of endpoint properties to domains of other entities through property maps. The term "flavor" or "instance" refers to an abstracted set of computing resources, with well-specified properties such as CPU, RAM and Storage. Thus, a flavor can be seen as an ANE with properties defined in terms of TIFSA. A flavor or instance is a group of 1 or more elements that can be reached via one or more network addresses. So an instance can also be seen as a PID that groups one or more IP addresses. In a context such as the one defined in CNTT, an ALTO property map could be used to expose TIFSA information of potential candidate flavors, i.e., potential NFVI-PoPs where an application or service can be deployed.

Figure 3 below depicts an example of a typical edge-cloud scenario [RFC9275] where each ANE represents a flavor/instance that resides on different cloud servers. Flavors on the "on-premise" edge nodes have

limited resources (but are closer to the end hosts), and flavors on the site-radio edge node and access central office (CO) have more available resources.

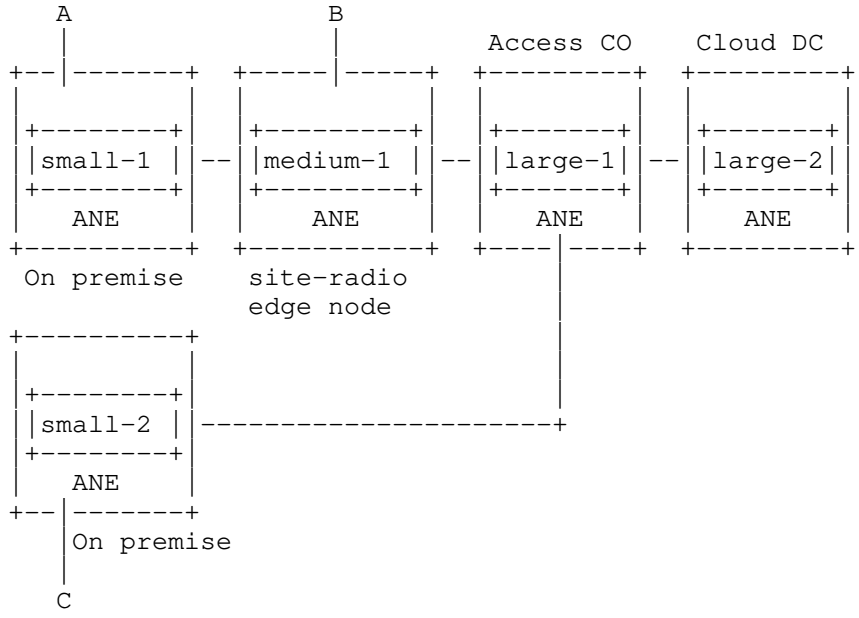


Figure 3: Example Use Case for Service Edge.

Based on the reference scenario (Figure 3), Figure 4 shows fictitious TIFSA property types for entities of domain type "ane":

flavor -name	type (T)	interface (I)	f-cpu (F)	f-ram (F)	f-disk (F)	f-bw (F)	S	A
small-1	basic	1 Gbps	1	512 MB	1 GB	1 Gbps		
small-2	network-intensive	9 Gbps	1	512 MB	1 GB	1 Gbps		
medium-1	network-intensive	25 Gbps	2	4 GB	40 GB	1 Gbps		
large-1	compute-intensive	50 Gbps	4	8 GB	80 GB	1 Gbps		
large-2	compute-intensive	100 Gbps	8	16 GB	160 GB	1 Gbps		

Figure 4: ALTO Property Map.

Subsequently, an ALTO client may request flavor(s) information from source [A] to destinations [B,C]. The following is a simplified example of an ALTO client request and the corresponding response. Note that the response consists of two parts: (i) ANE array for each source and destination pair (out of the scope of this document), and (ii) the requested properties of ANEs.

```
POST /costmap/pv HTTP/1.1
Host: alto.example.com
Accept: multipart/related;type=application/alto-costmap+json,
        application/alto-error+json
Content-Length: 163
Content-Type: application/alto-costmapfilter+json

{
  "cost-type": {
    "cost-mode": "array",
    "cost-metric": "ane-path"
  },
  "pids": {
    "srcs": [ "A" ],
    "dsts": [ "B", "C" ]
  },
  "ane-property-names": [
    "type",
```

```
      "cpu",
      "ram",
      "disk"
    ]
  }

  HTTP/1.1 200 OK
  Content-Length: 952
  Content-Type: multipart/related; boundary=example-1;
                type=application/alto-costmap+json

  Content-ID: <costmap@alto.example.com>
  Content-Type: application/alto-costmap+json

  {
    "meta": {
      "vtag": {
        "resource-id": "filtered-cost-map-pv.costmap",
        "tag": "d827f484cb66ce6df6b5077cb8562b0a"
      },
      "dependent-vtags": [
        {
          "resource-id": "my-default-networkmap",
          "tag": "c04bc5da49534274a6daeee8ea1dec62"
        }
      ],
      "cost-type": {
        "cost-mode": "array",
        "cost-metric": "ane-path"
      }
    },
    "cost-map": {
      "A": {
        "B": [ "small-1", "medium-1"],
        "C": [ "small-1", "medium-1", "large-1", "small-2" ]
      }
    }
  }

  Content-ID: <propmap@alto.example.com>
  Content-Type: application/alto-propmap+json

  {
      "meta" : {
      },
      "property-map": {
        ".ane:small-1":
          {"type" : "basic", "cpu" : 1,
```

```
            "ram" : "512MB", "disk" : 1GB},
          ".ane:small-2":
            {"type" : "network-intensive", "cpu" : 1,
              "ram" : "512MB", "disk" : 1GB},
          ".ane:medium-1":
            {"type" : "compute-intensive", "cpu" : 2,
              "ram" : "4GB", "disk" : 40GB},
          ".ane:large-1":
           {"type" : "compute-intensive", "cpu" : 4,
             "ram" : "8GB", "disk" : 80GB}
      }    }
```

6. Use Cases

6.1. Open Abstraction for Edge Computing

As shown in this document, modern applications such as AR/VR, V2X, or IoT, require bringing compute closer to the edge in order to meet strict bandwidth, latency, and jitter requirements. While this deployment process resembles the path taken by the main cloud providers (notably, AWS, Facebook, Google and Microsoft) to deploy their large-scale datacenters, the edge presents a key difference: datacenter clouds (both in terms of their infrastructure and the applications run by them) are owned and managed by a single organization, whereas edge clouds involve a complex ecosystem of operators, vendors, and application providers, all striving to provide a quality end-to-end solution to the user. This implies that, while the traditional cloud has been implemented for the most part by using vertically optimized and closed architectures, the edge will necessarily need to rely on a complete ecosystem of carefully designed open standards to enable horizontal interoperability across all the involved parties. This document envisions ALTO playing a role as part of the ecosystem of open standards that are necessary to deploy and operate the edge cloud.

As an example, consider a user of an XR application who arrives at his/her home by car. The application runs by leveraging compute capabilities from both the car and the public 5G edge cloud. As the user parks the car, 5G coverage may diminish (due to building interference) making the home local Wi-Fi connectivity a better choice. Further, instead of relying on computational resources from the car and the 5G edge cloud, latency can be reduced by leveraging computing devices (PCs, laptops, tablets) available from the home edge cloud. The application’s decision to switch from one domain to another, however, demands knowledge about the compute and communication resources available both in the 5G and the Wi-Fi domains, therefore requiring interoperability across multiple industry standards (for instance, IETF and 3GPP on the public side,

and IETF and LF Edge [LF-EDGE] on the private home side). ALTO can be positioned to act as an abstraction layer supporting the exposure of communication and compute information independently of the type of domain the application is currently residing in.

Future versions of this document will elaborate further on this use case.

6.2. Optimized placement of microservice components

Current applications are transitioning from a monolithic service architecture towards the composition of microservice components, following cloud-native trends. The set of microservices can have associated SLOs which impose constraints not only in terms of required compute resources (CPU, storage, ...) dependent on the compute facilities available, but also in terms of performance indicators such as latency, bandwidth, etc, which impose restrictions in the networking capabilities connecting the computing facilities. Even more complex constrains, such as affinity among certain microservices components could require complex calculations for selecting the most appropriate compute nodes taken into consideration both network and compute information.

Thus, service/application orchestrators can benefit from the information exposed by ALTO at the time of deciding the placement of the microservices in the network.

6.3. Distributed AI Workloads

Generative AI is a technological feat that opens up many applications such as holding conversations, generating art, developing a research paper, or writing software, among many others. Yet this innovation comes with a high cost in terms of processing and power consumption. While data centers are already running at capacity, it is projected that transitioning current search engine queries to leverage generative AI will increase costs by 10 times compared to traditional search methods [DC-AI-COST]. As (1) computing nodes (CPUs and GPUs) are deployed to build the edge cloud through technologies like 5G and (2) with billions of mobile user devices globally providing a large untapped computational platform, shifting part of the processing from the cloud to the edge becomes a viable and necessary step towards enabling the AI-transition. There are at least four drivers supporting this trend:

* Computational and energy savings: Due to savings from not needing large-scale cooling systems and the high performance-per-watt efficiency of the edge devices, some workloads can run at the edge at a lower computational and energy cost [EDGE-ENERGY], especially when considering not only processing but also data transport.

* Latency: For applications such as driverless vehicles which require real-time inference at very low latency, running at the edge is necessary.

* Reliability and performance: Peaks in cloud demand for generative AI queries can create large queues and latency, and in some cases even lead to denial of service. In some cases, limited or no connectivity requires running the workloads at the edge.

* Privacy, security, and personalization: A "private mode" allows users to strictly utilize on-device (or near-the-device) AI to enter sensitive prompts to chatbots, such as health questions or confidential ideas.

These drivers lead to a distributed computational model that is hybrid: Some AI workloads will fully run in the cloud, some will fully run in the edge, and some will run both in the edge and in the cloud. Being able to efficiently run these workloads in this hybrid, distributed, cloud-edge environment is necessary given the aforementioned massive energy and computational costs. To make optimized service and workload placement decisions, information about both the compute and communication resources available in the network is necessary too.

Consider as an example a large language model (LLM) used to generate text and hold intelligent conversations. LLMs produce a single token per inference, where a token is almost equivalent to a word. Pipelining and parallelization techniques are used to optimize inference, but this means that a model like GPT-3 could potentially go through all 175 billion parameters that are part of it to generate a single word. To efficiently run these computational-intensive workloads, it is necessary to know the availability of compute resources in the distributed system. Suppose that a user is driving a car while conversing with an AI model. The model can run inference on a variety of compute nodes, ordered from lower to higher compute power as follows: (1) the user's phone, (2) the computer in the car, (3) the 5G edge cloud, and (4) the datacenter cloud. Correspondingly, the system can deploy four different models with different levels of precision and compute requirements. The simplest model with the least parameters can run in the phone, requiring less compute power but yielding lower accuracy. Three other models ordered in increasing value of accuracy and computational complexity

can run in the car, the edge, and the cloud. The application can identify the right trade-off between accuracy and computational cost, combined with metrics of communication bandwidth and latency, to make the right decision on which of the four models to use for every inference request. Note that this is similar to the resolution/ bandwidth trade-off commonly found in the image encoding problem, where an image can be encoded and transmitted at different levels of resolution depending on the available bandwidth in the communication channel. In the case of AI inference, however, not only bandwidth is a scarce resource, but also compute. ALTO extensions to support the exposure of compute resources would allow applications to make optimized decisions on selecting the right computational resource, supporting the efficient execution of hybrid AI workloads.

7. Security Considerations

TODO Security

8. IANA Considerations

This document has no IANA actions.

9. Conclusions

Telco networks will increasingly contain a number of interconnected data centers and edge clouds of different sizes and characteristics, allowing flexibility in the dynamic deployment of functions and applications for advanced services. The overall objective of this document is to begin a discussion in the ALTO WG regarding the suitability of the ALTO protocol for determining where to deploy a function or application in these distributed computing environments. The result of these discussions will be reflected in future versions of this draft.

10. References

10.1. Normative References

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <https://www.rfc-editor.org/info/rfc7285>.

[RFC7752] Gredler, H., Ed., Medved, J., Previdi, S., Farrel, A., and S. Ray, "North-Bound Distribution of Link-State and Traffic Engineering (TE) Information Using BGP", RFC 7752, DOI 10.17487/RFC7752, March 2016, <https://www.rfc-editor.org/info/rfc7752>.

[RFC9275] Gao, K., Lee, Y., Randriamasy, S., Yang, Y., and J. Zhang, "An Extension for Application-Layer Traffic Optimization (ALTO): Path Vector", RFC 9275, DOI 10.17487/RFC9275, September 2022, <https://www.rfc-editor.org/info/rfc9275>.

[RFC9240] Roome, W., Randriamasy, S., Yang, Y., Zhang, J., and K. Gao, "An Extension for Application-Layer Traffic Optimization (ALTO): Entity Property Maps", RFC 9240, DOI 10.17487/RFC9240, July 2022, <https://www.rfc-editor.org/info/rfc9240>.

[I-D.ietf-alto-performance-metrics] Wu, Q., Yang, Y. R., Lee, Y., Dhody, D., Randriamasy, S., and L. M. Contreras, "ALTO Performance Cost Metrics", Work in Progress, Internet-Draft, draft-ietf-alto-performance-metrics-28, 21 March 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-alto-performance-metrics-28>.

[I-D.ietf-teas-sf-aware-topo-model] Bryskin, I., Liu, X., Lee, Y., Guichard, J., Contreras, L. M., Ceccarelli, D., Tantsura, J., and D. Shytyi, "SF Aware TE Topology YANG Model", Work in Progress, Internet-Draft, draft-ietf-teas-sf-aware-topo-model-11, 12 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-teas-sf-aware-topo-model-11>.

[I-D.llc-teas-dc-aware-topo-model] Lee, Y., Liu, X., and L. M. Contreras, "DC aware TE topology model", Work in Progress, Internet-Draft, draft-llc-teas-dc-aware-topo-model-03, 10 July 2023, <https://datatracker.ietf.org/doc/html/draft-llc-teas-dc-aware-topo-model-03>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

10.2. Informative References

[CNTT] "Cloud Infrastructure Telco Taskforce Reference Model", https://github.com/cntt-n/CNTT/wiki , June 2022.

[GSMA] "Cloud Infrastructure Reference Model, Version 2.0", https://www.gsma.com/newsroom/wp-content/uploads//NG.126-v2.0-1.pdf , October 2021.

[Anuket] "Anuket Project", https://wiki.anuket.io/ , October 2022.

[LF-EDGE] "Linux Foundation Edge", https://www.lfedge.org/ , March 2023.

[EDGE-ENERGY] "Estimating energy consumption of cloud, fog, and edge computing infrastructures", IEEE Transactions on Sustainable Computing , 2019.

[DC-AI-COST] "Generative AI Breaks The Data Center - Data Center Infrastructure And Operating Costs Projected To Increase To Over $76 Billion By 2028", Forbes, Tirias Research Report , 2023.

Acknowledgments

The work of Luis M. Contreras has been partially funded by the European Union under the Horizon Europe project CODECO (COgnitive, Decentralised Edge- Cloud Orchestration), grant number 101092696.

Authors' Addresses

Luis M. Contreras
Telefonica
Email: luismiguel.contrerasmurillo@telefonica.com

Sabine Randriamasy
Nokia Bell Labs
Email: sabine.randriamasy@nokia-bell-labs.com

Jordi Ros-Giralt
Qualcomm Europe, Inc.
Email: jros@qti.qualcomm.com

Danny Lachos
Benocs
Email: dlachos@benocs.com

Christian Rothenberg
Unicamp
Email: chesteve@dca.fee.unicamp.br

ALTO Working Group
Internet-Draft
Intended status: Standards Track
Expires: 12 January 2023

R. Schott
Deutsche Telekom
Y. Yang
Yale University
K. Gao
Sichuan University
J. Zhang
Tongji University

11 July 2022

ALTO/H2: The ALTO Protocol using HTTP/2
draft-ietf-alto-new-transport-01

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 12 January 2023.

Table of Contents

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.

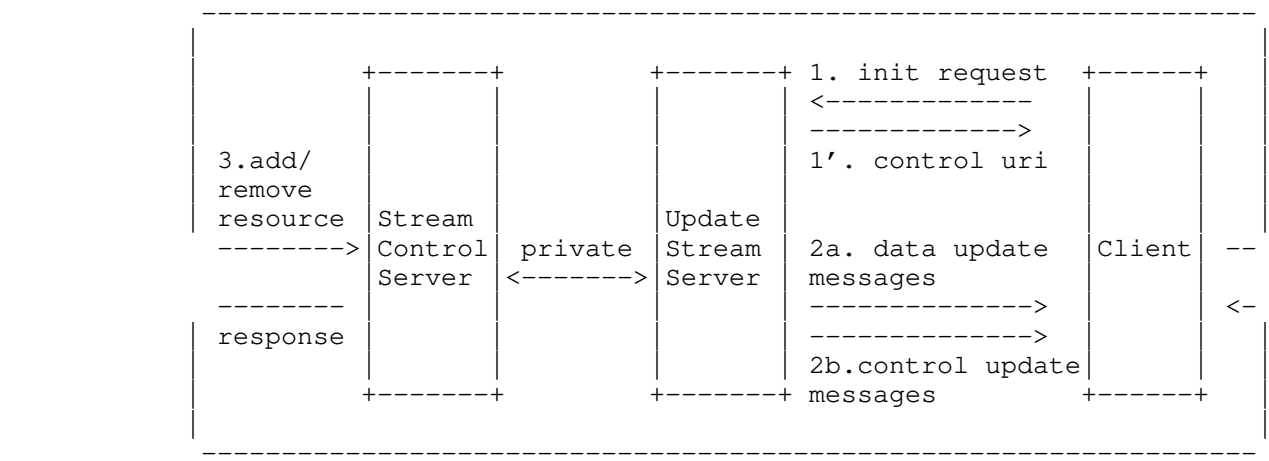


Figure 1: ALTO SSE Architecture and Message Flow.

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 the 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 transfers and respects HTTP/2 semantics such as the semantics of PUSH_PROMISE.

To allow flexible deployment, the new transport protocol should be flexible, in particular,

* R7: The design should support capability negotiation.

3. ALTO/H2 Design Overview

A key design of ALTO/H2 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.

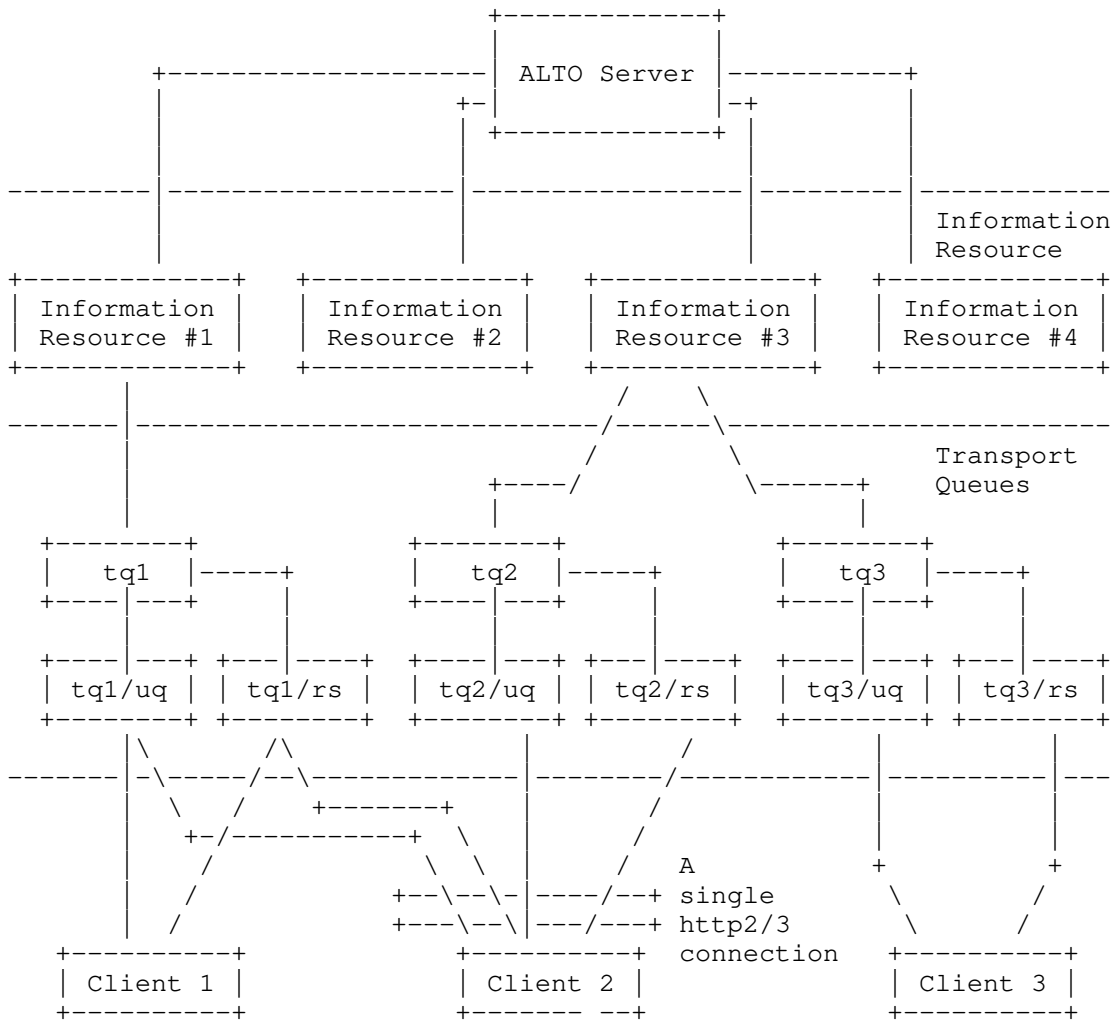
* Each transport queue maintains two states: (1) the incremental update message queue, which includes a sequence of incremental update messages and (2) the receiver set, which includes the set of receivers receiving incremental push updates from the ALTO server.

* 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, or Client 3) maintains a single HTTP/2 connection with the ALTO server.

Information Resource:

a) Static resource (#1) such as NetworkMap
b) Filterable resource (#3) such as FilteredCostMap



tqi = transport queue i
tqi/uq = incremental updates queue of transport queue i
tqi/rs = receiver set of transport queue i

Figure 2: ALTO New Transport Information Structure.

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
  // pull mode
  Client requests transport queue status of tq
  Client requests an element in the incremental update queue

  // push mode
  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:

```
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
 { "uq":
    [
      {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 = /tqs/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:        "8eb1d4785acd42231bfecdf0222x59740b0b2e3f",
       "link":       "/tqs/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 a sequence number 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
 {
   "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 = /tqs/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.

Creat: A client can add itself in the receiver set by using the HTTP PUT method: PUT 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" error) 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: the 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 from the server to itself 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 after B)

* Encode dependency to enforce semantics (correctness)

To realize the objectives, 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 bu 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-transport+json",
      "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-transport+json". 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 application/alto-transport+json 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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <https://www.rfc-editor.org/info/rfc7285>.

[RFC7540] Belshe, M., Peon, R., and M. Thomson, "Hypertext Transfer Protocol Version 2 (HTTP/2)", RFC 7540, DOI 10.17487/RFC7540, May 2015, <https://www.rfc-editor.org/info/rfc7540>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8895] Roome, W. and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Incremental Updates Using Server-Sent Events (SSE)", RFC 8895, DOI 10.17487/RFC8895, November 2020, <https://www.rfc-editor.org/info/rfc8895>.

13.2. Informative References

[RFC7971] Stiemerling, M., Kiesel, S., Scharf, M., Seidel, H., and S. Previdi, "Application-Layer Traffic Optimization (ALTO) Deployment Considerations", RFC 7971, DOI 10.17487/RFC7971, October 2016, <https://www.rfc-editor.org/info/rfc7971>.

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

ALTO
Internet-Draft
Intended status: Standards Track
Expires: 6 July 2024

K. Gao
Sichuan University
R. Schott
Deutsche Telekom
Y. R. Yang
L. Delwiche
L. Keller
Yale University
3 January 2024

The ALTO Transport Information Publication Service
draft-ietf-alto-new-transport-22

Abstract

The ALTO Protocol (RFC 7285) leverages HTTP/1.1 and is designed for the simple, sequential request-reply use case, in which an ALTO client requests a sequence of information resources and the server responds with the complete content of each resource one at a time.

ALTO incremental updates using Server-Sent Events (SSE) (RFC 8895) defines a multiplexing protocol on top of HTTP/1.x, so that an ALTO server can incrementally push resource updates to clients whenever monitored network information resources change, allowing the clients to monitor multiple resources at the same time. However, HTTP/2 and later versions already support concurrent, non-blocking transport of multiple streams in the same HTTP connection.

To take advantage of newer HTTP features, this document introduces the ALTO Transport Information Publication Service (TIPS). TIPS uses an incremental RESTful design to give an ALTO client the new capability to explicitly, concurrently (non-blocking) request (pull) specific incremental updates using native HTTP/2 or HTTP/3, while still functioning for HTTP/1.1.

Discussion Venues

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

Discussion of this document takes place on the Application-Layer Traffic Optimization 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/ietf-wg-alto/draft-ietf-alto-new-transport.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Application-Layer Traffic Optimization (ALTO) provides means for network applications to obtain network status information. So far, the ALTO information can be transported in two ways:

1. The ALTO base protocol [RFC7285], which is designed for the simple use case in which an ALTO client requests a network information resource, and the server sends the complete content of the requested information (if any) resource to the client.

2. ALTO incremental updates using Server-Sent Events (ALTO/SSE) [RFC8895], which is designed for an ALTO client to indicate to the server that it wants to receive updates for a set of resources and the server can then concurrently and incrementally push updates to that client whenever monitored resources change.

Both protocols are designed for HTTP/1.1 [RFC9112], but HTTP/2 [RFC9113] and HTTP/3 [RFC9114] can support HTTP/1.1 workflows. However, HTTP/2 and HTTP/3 provide features that can improve certain properties of ALTO and ALTO/SSE.

* First, consider the ALTO base protocol, which is designed to transfer only complete information resources. A client can run the base protocol on top of HTTP/2 or HTTP/3 to request multiple information resources in concurrent streams, but each request must be for a complete information resource: there is no capability for the server to transmit incremental updates. Hence, there can be a large overhead when the client already has an information resource and then there are small changes to the resource.

* Next, consider ALTO/SSE [RFC8895]. Although ALTO/SSE can transfer incremental updates, it introduces a customized multiplexing protocol on top of HTTP, assuming a total-order message channel from the server to the client. The multiplexing design does not provide naming (i.e., a resource identifier) to individual incremental updates. Such a design cannot use concurrent data streams available in HTTP/2 and HTTP/3, because both cases require a resource identifier. Additionally, ALTO/SSE is a push-only protocol, which denies the client flexibility in choosing how and when it receives updates.

To mitigate these concerns, this document introduces a new ALTO service called the Transport Information Publication Service (TIPS). TIPS uses an incremental RESTful design to provide an ALTO client with a new capability to explicitly, concurrently issue non-blocking requests for specific incremental updates using native HTTP/2 or HTTP/3, while still functioning for HTTP/1.1.

While ALTO/SSE [RFC8895] and TIPS both can transport incremental updates of ALTO information resources to clients, they have different design goals. The TIPS extension enables more scalable and robust distribution of incremental updates, but is missing the session management and built-in server push capabilities of ALTO/SSE. From

the performance perspective, TIPS is optimizing throughput by leveraging concurrent and out-of-order transport of data, while ALTO/SSE is optimizing latency as new events can be immediately transferred to the clients without waiting for another round of communication when there are multiple updates. Thus, we do not see TIPS as a replacement but as a complement of ALTO/SSE. One example of combining these two extensions is as shown in Section 6.3.3.

Note that future extensions may leverage server push, a feature of HTTP/2 [RFC9113] and HTTP/3 [RFC9114], as an alternative of SSE. We discuss why this alternative design is not ready in Appendix C.

Specifically, this document specifies:

* Extensions to the ALTO Protocol for dynamic subscription and efficient uniform update delivery of an incrementally changing network information resource.

* A new resource type that indicates the TIPS updates graph model for a resource.

* URI patterns to fetch the snapshots or incremental updates.

Some operational complexities that must be taken into consideration when implementing this extension are discussed in Section 8, including load balancing Section 8.1, fetching and processing incremental updates of dependent resources Section 8.2

Appendix B discusses to what extent the TIPS design adheres to the Best Current Practices for building protocols with HTTP [RFC9205].

1.1. 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.

1.2. Notations

This document uses the same syntax and notations as introduced in Section 8.2 of [RFC7285] to specify the extensions to existing ALTO resources and services.

2. TIPS Overview

2.1. Transport Requirements

The ALTO Protocol and its extensions support two transport mechanisms: First, a client can directly request an ALTO resource and obtain a complete snapshot of that ALTO resource, as specified in the base protocol [RFC7285]; Second, a client can subscribe to incremental changes of one or multiple ALTO resources using the incremental update extension [RFC8895], and a server pushes the updates to the client through Server Sent Events (SSE).

However, the current transport mechanisms are not optimized for storing, transmitting, and processing (incremental) updates of ALTO information resources. Specifically, the new transport mechanism must satisfy the following requirements:

Incremental updates: Incremental updates only maintain and transfer the "diff" upon changes. Thus, it is more efficient than storing and transferring the full updates, especially when the change of an ALTO resource is minor. The base protocol does not support incremental updates and the current incremental update mechanism in [RFC8895] has limitations (as discussed below).

Concurrent, non-blocking update transmission: When a client needs to receive and apply multiple incremental updates, it is desired to transmit the updates concurrently to fully utilize the bandwidth and to reduce head-of-line blocking. The ALTO incremental update extension [RFC8895], unfortunately, does not satisfy this requirement -- even though the updates can be multiplexed by the server to avoid head-of-line blocking between multiple resources, the updates are delivered sequentially and can suffer from head-of-line blocking inside the connection, for example, when there is a packet loss.

Long-polling updates: Long-polling updates can reduce the time to send the request, making it possible to achieve sub-RTT transmission of ALTO incremental updates. In [RFC8895], this requirement is fulfilled using server-sent event (SSE) and is still desired in the ALTO new transport.

Backward compatibility: While some of the previous requirements are offered by HTTP/2 [RFC9113] and HTTP/3 [RFC9114], it is desired that the ALTO new transport mechanism can work with HTTP/1.1 as many development tools and current ALTO implementations are based on HTTP/1.1.

The ALTO new transport specified in this document satisfies all the design requirements:

* This document reuses the data format introduced in [RFC8895] that enables incremental updates using JSON patches or merge patches.

* This document introduce a unified data model to describe the changes (snapshots and incremental updates) of an ALTO resource, referred to as a TIPS view. In the data model, snapshots and incremental updates are indexed as individual HTTP resources following a unified naming convention, independent of the HTTP version. Thus, these updates can be concurrently requested and be transferred in a non-blocking manner either by using multiple connections or leveraging multiplexed data transfer offered by HTTP/2 or HTTP/3.

* The unified naming convention is based on a monotonically increasing sequence number, making it possible for a client to construct the URL of a future update and send a long-polling request.

* The unified naming convention is independent of the HTTP versions and can operate atop HTTP/1.1, HTTP/2 or HTTP/3.

This document assumes the deployment model discussed in Appendix A.

2.2. TIPS Terminology

In addition to the terms defined in [RFC7285], this document uses the following terms:

Transport Information Publication Service (TIPS): Is a new type of ALTO service, as specified in this document, to enable a uniform transport mechanism for updates of an incrementally changing ALTO network information resource.

Network information resource: Is a piece of retrievable information about network state, per [RFC7285].

TIPS view (tv): Is defined in this document to be the container of incremental transport information about the network information resource. The TIPS view has one basic component, updates graph (ug), but may include other transport information.

Updates graph (ug): Is a directed, acyclic graph whose nodes represent the set of versions of an information resource, and edges the set of update items to compute these versions. An ALTO map service (e.g., Cost Map, Network Map) may need only a single

updates graph. A dynamic network information service (e.g., Filtered Cost Map) may create an updates graph (within a new TIPS view) for each unique request. Encoding of a updates graph is specified in Section 6.1.

Version: Represents a historical content of an information resource. For an information resource, each version is associated with and uniquely identified by a monotonically and consecutively increased sequence number. This document uses the term "version s" to refer to the version associated with sequence number "s". Version is encoded as a JSONNumber, as specified in Section 6.1.

Start sequence number (start-seq): Is the smallest non-zero sequence number in an updates graph.

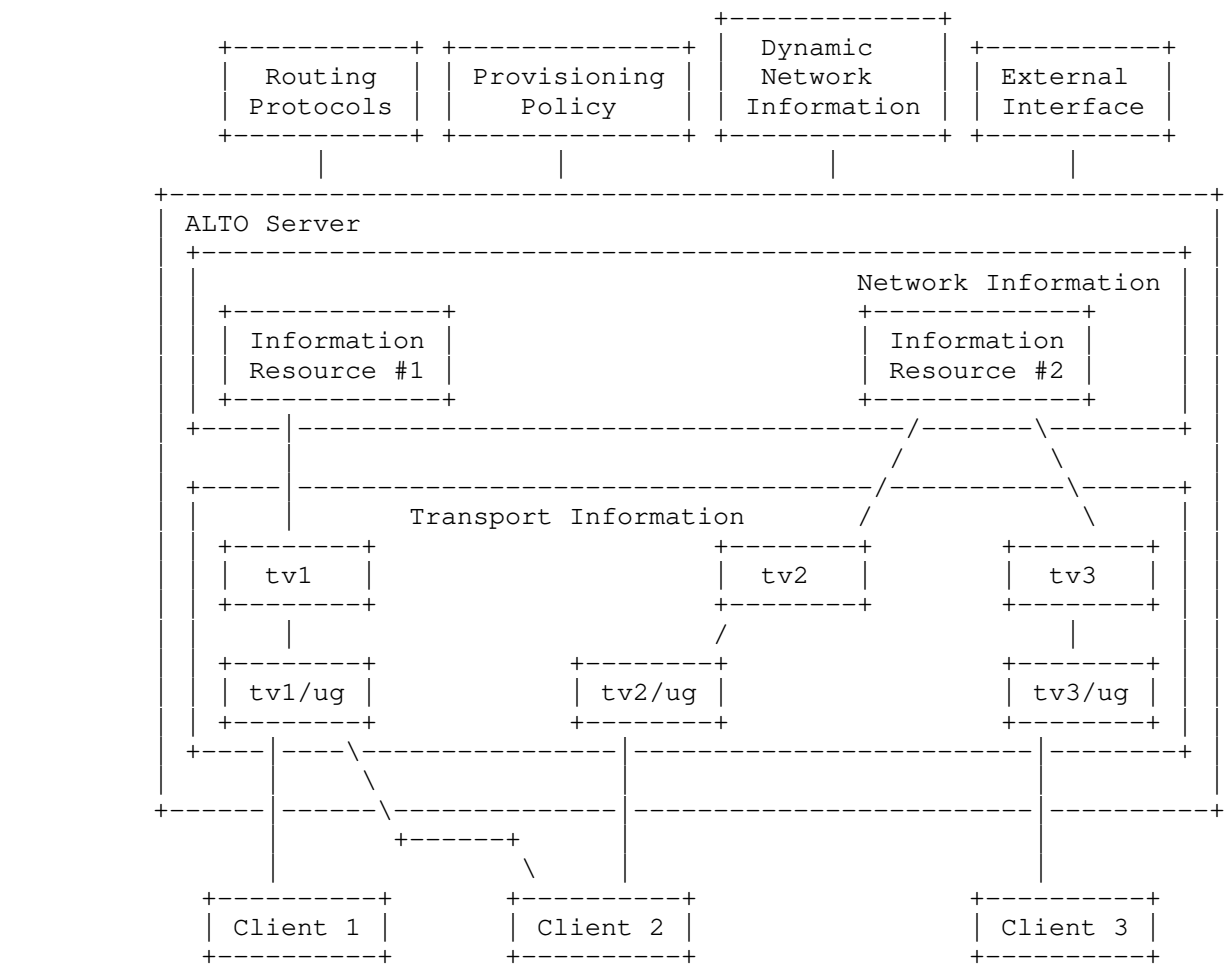
End sequence number (end-seq): Is the largest sequence number in an updates graph.

Snapshot: Is a full replacement of a resource and is contained within an updates graph.

Incremental update: Is a partial replacement of a resource contained within an updates graph, codified in this document as a JSON Merge Patch or JSON Patch. An incremental update is mandatory if the source version (i) and target version (j) are consecutive, i.e., i + 1 = j, and optional or a shortcut otherwise. Mandatory incremental updates are always in an updates graph, while optional/shortcut incremental updates may or may not be included in an updates graph.

Update item: Refers to the content on an edge of the updates graph, which can be either a snapshot or an incremental update. An update item can be considered as a pair (op, data) where op denotes whether the item is an incremental update or a snapshot, and data is the content of the item.

ID#i-#j: Denotes the update item on a specific edge in the updates graph to transition from version i to version j, where i and j are the sequence numbers of the source node and the target node of the edge, respectively.



tvi = TIPS view i
tvi/ug = incremental updates graph associated with tvi

Figure 1: Overview of ALTO TIPS

Figure 1 shows an example illustrating an overview of the ALTO TIPS service. The server provides the TIPS service of two information resources (#1 and #2) where #1 is an ALTO map service, and #2 is a filterable service. There are 3 ALTO clients (Client 1, Client 2, and Client 3) that are connected to the ALTO server.

Each client uses the TIPS view to retrieve updates. Specifically, a TIPS view (tv1) is created for the map service #1, and is shared by multiple clients. For the filtering service #2, two different TIPS views (tv2 and tv3) are created upon different client requests with different filter sets.

3. TIPS Updates Graph

In order to provide incremental updates for a resource, an ALTO server creates an updates graph, which is a directed, acyclic graph that contains a sequence of incremental updates and snapshots (collectively called update items) of a network information resource.

3.1. Basic Data Model of Updates Graph

For each resource (e.g., a cost map, a network map), the incremental updates and snapshots can be represented using the following directed acyclic graph model, where the server tracks the change of the resource maps with version IDs that are assigned sequentially (i.e., incremented by 1 each time):

* Each node in the graph is a version of the resource, which is identified by a sequence number (defined as a JSONNumber). Version 0 is reserved as the initial state (empty/null).

* A tag identifies the content of a node. A tag has the same format as the "tag" field in Section 10.3 of [RFC7285] and is valid only within the scope of resource.

* Each edge is an update item. In particular, the edge from i to j is the update item to transit from version i to version j.

* Version is path-independent (different paths arrive at the same version/node has the same content)

A concrete example is shown in Figure 2. There are 7 nodes in the graph, representing 7 different versions of the resource. Edges in the figure represent the updates from the source version to the target version. Thick lines represent mandatory incremental updates (e.g., ID103-104), dotted lines represent optional incremental updates (e.g., ID103-105), and thin lines represent snapshots (e.g., ID0-103). Note that node content is path independent: the content of node v can be obtained by applying the updates from any path that ends at v. For example, assume the latest version is 105 and a client already has version 103. The base version of the client is 103 as it serves as a base upon which incremental updates can be applied. The target version 105 can either be directly fetched as a snapshot, computed incrementally by applying the incremental updates

between 103 and 104, then 104 and 105, or if the optional update from 103 to 105 exists, computed incrementally by taking the "shortcut" path from 103 to 105.

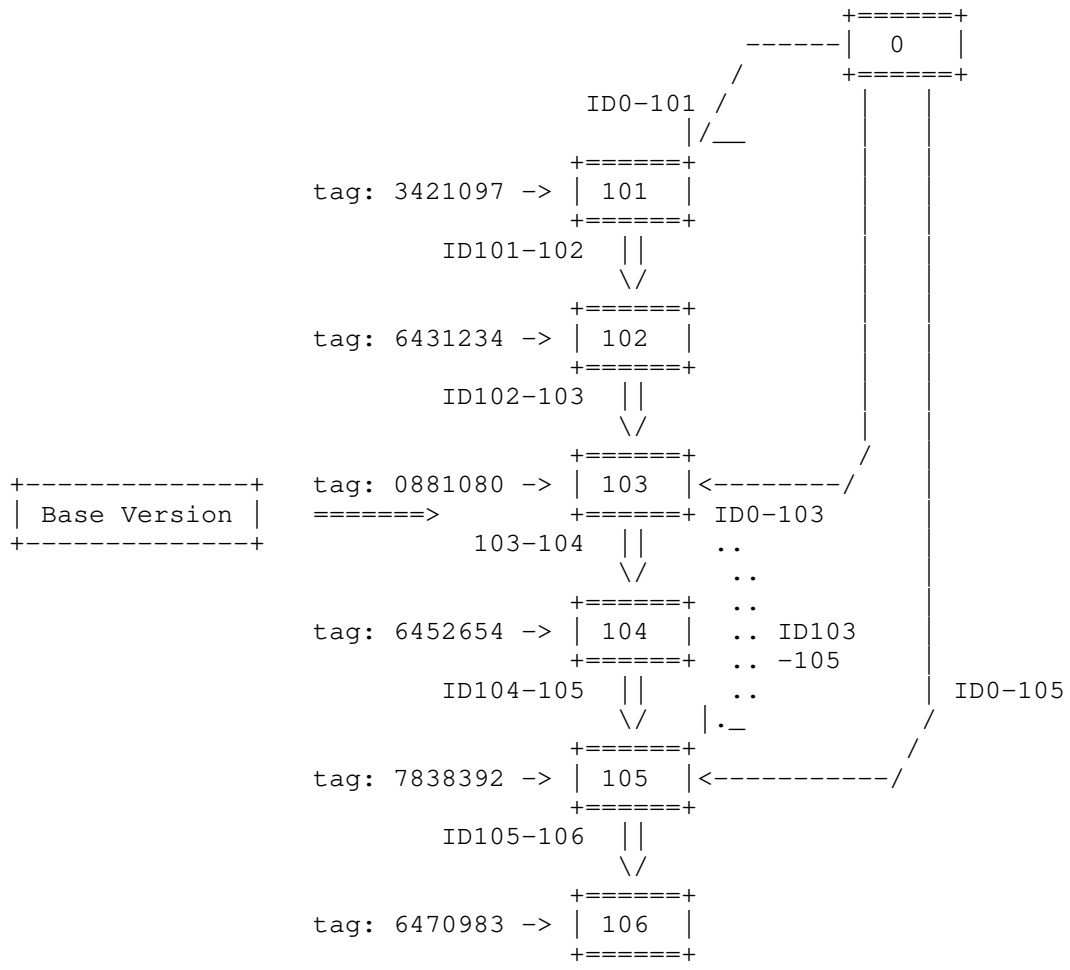


Figure 2: TIPS Model Example

3.2. Updates Graph Modification Invariants

A server may change its updates graph (to compact, to add nodes, etc.), but it must ensure that any resource state that it makes available is reachable by clients, either directly via a snapshot (that is, relative to 0) or indirectly by requesting an earlier snapshot and a contiguous set of incremental updates. Additionally, to allow clients to proactively construct URIs for future update

items, the ID of each added node in the updates graph must increment contiguously by 1. More specifically, the updates graph MUST satisfy the following invariants:

* Continuity: At any time, let ns denote the smallest non-zero version (i.e., start-seq) in the update graph and ne denote the latest version (i.e., end-seq). Then any version in between ns and ne MUST also exist. This implies that the incremental update from ni to ni + 1 exists for any ns <= ni <= ne, and all versions in the update graph (except 0) is an integer interval [ns, ne].

* Feasibility: Let ns denote the start-seq in the update graph. The server MUST provide a snapshot of ns and, in other words, there is always a direct link to ns in the update graph.

* "Right shift" only: Assume a server provides versions in [n1, n2] at time t and versions in [n1', n2'] at time t'. If t' > t, then n1' >= n1 and n2' >= n2.

For example, consider the case that a server compacts a resource's updates graph to conserve space, using the example model in Section 3.1. Assume at time 0, the server provides the versions {101, 102, 103, 104, 105, 106}. At time 1, both {103, 104, 105, 106} and {105, 106} are valid sets. However, {102, 103, 104, 105, 106} and {104, 105, 106} are not valid sets as there is no snapshot to version 102 or 104 in the update graph. Thus, there is a risk that the right content of version 102 (in the first example) or 104 (in the second example) cannot be obtained by a client that does not have the previous version 101 or 103, respectively.

4. TIPS Workflow and Resource Location Schema

4.1. Workflow

At a high level, an ALTO client first uses the TIPS service (denoted as TIPS-F and F is for frontend) to indicate the information resource(s) that the client wants to monitor. For each requested resource, the server returns a JSON object that contains a URI, which points to the root of a TIPS view (denoted as TIPS-V), and a summary of the current view, which contains the information to correctly interact with the current view. With the URI to the root of a TIPS view, clients can construct URIs (see Section 4.2) to fetch incremental updates.

An example workflow is shown in Figure 3. After the TIPS-F service receives the request from the client to monitor the updates of an ALTO resource, it creates a TIPS view service and returns the corresponding information to the client. The URI points to that

specific TIPS-V instance and the summary contains the start-seq and end-seq of the update graph, and a server-recommended edge to consume first, e.g., from i to j.

An ALTO client can then continuously pull each additional update with the information. For example, the client in Figure 3 first fetches the update from i to j, and then from j to j+1. Note that the update item at <tips-view-uri>/ug/<j>/<j+1> may not yet exist, so the server holds the request until the update becomes available (long polling).

A server MAY close a TIPS view at any time, e.g., under high system load or due to client inactivity. In the event that a TIPS view is closed, an edge request will receive error code 404 in response, and the client will have to request a new TIPS view URI.

If resources allow, a server SHOULD avoid closing TIPS views that have active polling edge requests or have recently served responses until clients have had a reasonable interval to request the next update, unless guided by specific control policies.

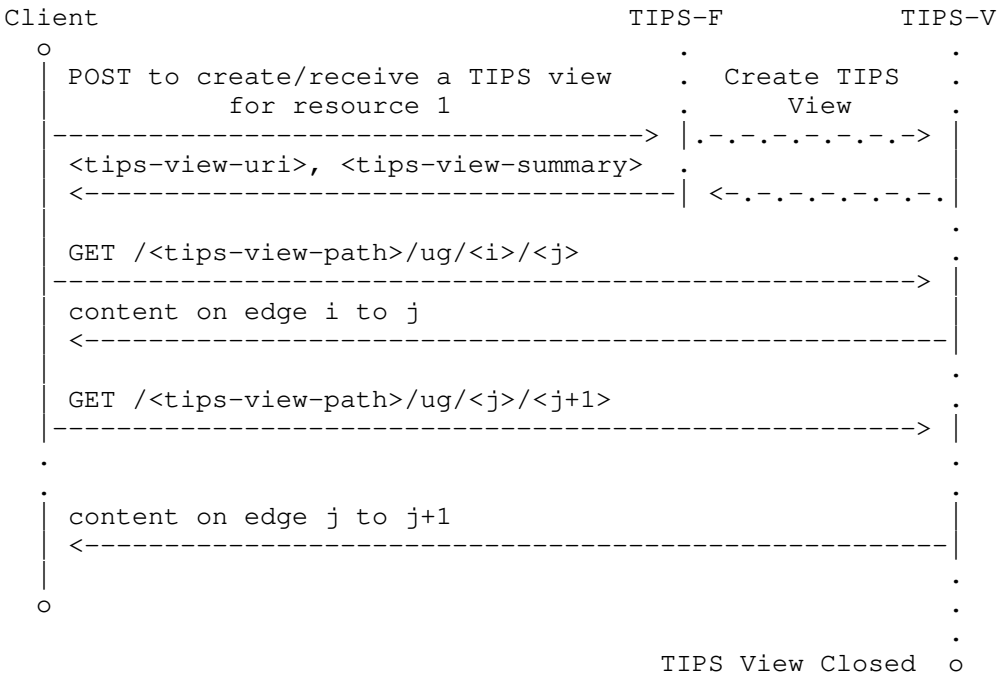


Figure 3: ALTO TIPS Workflow Supporting Client Pull

4.2. Resource Location Schema

The resource location schema defines how a client constructs URI to fetch incremental updates.

To access each update in an updates graph, consider the model represented as a "virtual" file system (adjacency list), contained within the root of a TIPS view URI (see Section 6.2 for the definition of tips-view-uri). For example, assuming that the update graph of a TIPS view is as shown in Figure 2, the location schema of this TIPS view will have the format as in Figure 4.

```
<tips-view-path>  // root path to a TIPS view
  |_ ug    // updates graph
  |   |_ 0
  |   |   |_ 101    // full 101 snapshot
  |   |   |_ 103
  |   |   \_ 105
  |   |_ 101
  |   |   \_ 102    // 101 -> 102 incremental update
  |   |_ 102
  |   |   \_ 103
  |   |_ 103
  |   |   |_ 104
  |   |   \_ 105    // optional shortcut 103 -> 105 incr. update
  |   |_ 104
  |   |   \_ 105
  |   \_ 105
  |       \_ 106
   \_ ...
```

Figure 4: Location Schema Example

TIPS uses this directory schema to generate template URIs which allow clients to construct the location of incremental updates after receiving the tips-view-uri from the server. The generic template for the location of the update item on the edge from node 'i' to node 'j' in the updates graph is:

```
<tips-view-uri>/ug/<i>/<j>
```

Due to the sequential nature of the update item IDs, a client can long poll a future update that does not yet exist (e.g., the incremental update from 106 to 107) by constructing the URI for the next edge that will be added, starting from the sequence number of the current last node (denoted as end-seq) in the graph to the next sequential node (with the sequence number of end-seq + 1):

```
<tips-view-uri>/ug/<end-seq>/<end-seq + 1>
```

Incremental updates of a TIPS view are read-only. Thus, they are fetched using the HTTP GET method.

5. TIPS Information Resource Directory (IRD) Announcement

To announce a TIPS information resource in the information resource directory (IRD), an ALTO server MUST specify the "media-type", "capabilities" and "uses" as follows.

5.1. Media Type

The media type of the Transport Information Publication Service resource is "application/alto-tips+json".

5.2. Capabilities

The capabilities field of TIPS is modeled on that defined in Section 6.3 of [RFC8895].

Specifically, the capabilities are defined as an object of type TIPSCapabilities:

```
object {
  IncrementalUpdateMediaTypes incremental-change-media-types;
} TIPSCapabilities;

object-map {
   ResourceID -> String;
} IncrementalUpdateMediaTypes;
```

Figure 5: TIPSCapabilities

with field:

incremental-change-media-types: If a TIPS can provide updates with incremental changes for a resource, the "incremental-change-media-types" field has an entry for that resource-id, and the value is the supported media types of incremental changes, separated by commas. For the implementation of this specification, this MUST be "application/merge-patch+json", "application/json-patch+json", or "application/merge-patch+json,application/json-patch+json", unless defined by a future extension.

When choosing the media types to encode incremental updates for a resource, the server MUST consider the limitations of the encoding. For example, when a JSON merge patch specifies that the

value of a field is null, its semantics are that the field is removed from the target and hence the field is no longer defined (i.e., undefined). This, however, may not be the intended result for the resource, when null and undefined have different semantics for the resource. In such a case, the server MUST choose JSON patch over JSON merge patch if JSON patch is indicated as a capability of the TIPS. If the server does not support JSON patch to handle such a case, the server then needs to send a full replacement.

5.3. Uses

The "uses" attribute MUST be an array with the resource-ids of every network information resource for which this TIPS can provide service.

This set MAY be any subset of the ALTO server's network information resources and MAY include resources defined in linked IRDs. However, it is RECOMMENDED that the ALTO server selects a set that is closed under the resource dependency relationship. That is, if a TIPS' "uses" set includes resource R1 and resource R1 depends on ("uses") resource R0, then the TIPS' "uses" set should include R0 as well as R1. For example, if a TIPS provides a TIPS view for a cost map, it should also provide a TIPS view for the network map upon which that cost map depends.

If the set is not closed, at least one resource R1 in the "uses" field of a TIPS depends on another resource R0 which is not in the "uses" field of the same TIPS. Thus, a client cannot receive incremental updates for R0 from the same TIPS service. If the client observes in an update of R1 that the version tag for R0 has changed, it must request the full content of R0, which is likely to be less efficient than receiving the incremental updates of R0.

5.4. An Example

Extending the IRD example in Section 8.1 of [RFC8895], Figure 6 is the IRD of an ALTO server supporting ALTO base protocol, ALTO/SSE, and ALTO TIPS.

```
  "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-network-map"],
    "capabilities": {
```

```
        "cost-type-names": ["num-routingcost"]
      }
    },
    "my-hopcount-map": {
      "uri": "https://alto.example.com/costmap/hopcount",
      "media-type": "application/alto-costmap+json",
      "uses": ["my-network-map"],
      "capabilities": {
        "cost-type-names": ["num-hopcount"]
      }
    },
    "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-network-map"],
      "capabilities": {
        "cost-type-names": ["num-routingcost", "num-hopcount"],
        "cost-constraints": false
      }
    },
    "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-tips": {
      "uri": "https://alto.example.com/updates-new/costs",
      "media-type": "application/alto-tips+json",
      "accepts": "application/alto-tipsparams+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",
          "my-simple-filtered-cost-map": "application/merge-patch+json"
        }
      }
    },
    "tips-sse": {
      "uri": "https://alto.example.com/updates/tips",
      "media-type": "text/event-stream",
      "accepts": "application/alto-updatestreamparams+json",
      "uses": [ "update-my-costs-tips" ],
      "capabilities": {
        "incremental-change-media-types": {
          "update-my-costs-tips": "application/merge-patch+json"
        }
      }
    }
```

Figure 6: Example of an ALTO Server Supporting ALTO Base Protocol, ALTO/SSE, and ALTO TIPS

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.

The resource "update-my-costs-tips" provides an ALTO TIPS service, and this is indicated by the media-type "application/alto-tips+json".

6. TIPS Management

Upon request, a server sends a TIPS view to a client. This TIPS view may be created at the time of the request or may already exist (either because another client has already created a TIPS view for the same requested network resource or because the server perpetually maintains a TIPS view for an often-requested resource).

6.1. Open Request

An ALTO client requests that the server provide a TIPS view for a given resource by sending an HTTP POST body with the media type "application/alto-tipsparams+json". That body contains a JSON object of type TIPSReq, where:

```
object {
   ResourceID   resource-id;
   [JSONString  tag;]
   [Object      input;]
} TIPSReq;
```

Figure 7: TIPSReq

with the following fields:

resource-id: The resource-id of an ALTO resource and MUST be in the TIPS' "uses" list (Section 5). If a client does not support all incremental methods from the set announced in the server's capabilities, the client MUST NOT use the TIPS service.

tag: If the resource-id is a GET-mode resource with a version tag (or "vtag"), as defined in Section 10.3 of [RFC7285], and the ALTO client has previously retrieved a version of that resource from ALTO, the ALTO client MAY set the "tag" field to the tag part of the client's version of that resource. The server MAY use the tag when calculating a recommended starting edge for the client to consume. Note that the client MUST support all incremental methods from the set announced in the server's capabilities for this resource.

input: If the resource is a POST-mode service that requires input, the ALTO client MUST set the "input" field to a JSON object with the parameters that the resource expects.

6.2. Open Response

The response to a valid request MUST be a JSON object of type AddTIPSResponse, denoted as media type "application/alto-tips+json":

```
object {
  URI               tips-view-uri;
  TIPSViewSummary   tips-view-summary;
} AddTIPSResponse;

object {
  UpdatesGraphSummary   updates-graph-summary;
} TIPSViewSummary;

object {
  JSONNumber       start-seq;
  JSONNumber       end-seq;
  StartEdgeRec     start-edge-rec;
} UpdatesGraphSummary;

object {
  JSONNumber       seq-i;
  JSONNumber       seq-j;
} StartEdgeRec;
```

Figure 8: AddTIPSResponse

with the following fields:

tips-view-uri: URI to the requested TIPS view. The value of this field MUST have the following format:

```
scheme "://" tips-view-host "/" tips-view-path

tips-view-host = host [ ":" port]
tips-view-path = path
```

where scheme MUST be "http" or "https" unless specified by a future extension, and host, port and path are as specified in Sections 3.2.2, 3.2.3, and 3.3 in [RFC3986]. An ALTO server SHOULD use the "https" scheme unless the contents of the TIPS view are intended to be publicly accessible and does not raise security concerns. The field MUST contain only ASCII characters. In case the original URL contains international characters (e.g., in the domain name), the ALTO server implementation MUST properly encode the URL into the ASCII format (e.g., using the "urlencode" function).

A server MUST NOT use the same URI for different TIPS views, either for different resources or different request bodies to the same resource. URI generation is implementation specific, for example, one may compute a Universally Unique Identifier (UUID, [RFC4122]) or a hash value based on the request, and append it to

a base URL. For performance considerations, it is NOT RECOMMENDED to use properties that are not included in the request body to determine the URI of a TIPS view, such as cookies or the client's IP address, which may result in duplicated TIPS views in cases such as mobile clients. However, this is not mandatory as a server may intentionally use client information to compute the TIPS view URI to provide service isolation between clients.

tips-view-summary: Contains an updates-graph-summary.

The updates-graph-summary field contains the starting sequence number (start-seq) of the updates graph and the last sequence number (end-seq) that is currently available, along with a recommended edge to consume (start-edge-rec). If the client does NOT provide a version tag, the server MUST recommend the edge of the latest snapshot available. If the client does provide a version tag, the server MUST either recommend the first incremental update edge starting from the client's tagged version or the edge of the latest snapshot. Which edge is selected depends on the implementation. For example, a server MAY calculate the cumulative size of the incremental updates available from that version onward and compare it to the size of the complete resource snapshot. If the snapshot is bigger, the server recommends the first incremental update edge starting from the client's tagged version. Otherwise, the server recommends the latest snapshot edge.

If the request has any errors, the TIPS service MUST return an HTTP "400 Bad Request" to the ALTO client; the body of the response follows the generic ALTO error response format specified in Section 8.5.2 of [RFC7285]. Hence, an example ALTO error response has the format shown in Figure 9.

```
HTTP/1.1 400 Bad Request
Content-Length: 131
Content-Type: application/alto-error+json

{
    "meta":{
        "code":  "E_INVALID_FIELD_VALUE",
        "field": "resource-id",
        "value": "my-network-map/#"
    }
}
```

Figure 9: ALTO Error Example

Note that "field" and "value" are optional fields. If the "value" field exists, the "field" field MUST exist.

* If the TIPS request does not have a "resource-id" field, the error code of the error message MUST be E_MISSING_FIELD and the "field" field, if present, MUST be "resource-id". The TIPS service MUST NOT create any TIPS view.

* If the "resource-id" field is invalid or is not associated with the TIPS, the error code of the error message MUST be E_INVALID_FIELD_VALUE. If present, the "field" field MUST be the full path of the "resource-id" field, and the "value" field MUST be the invalid resource-id.

* If the resource is a POST-mode service that requires input, the client MUST set the "input" field to a JSON object with the parameters that that resource expects. If the "input" field is missing or invalid, TIPS MUST return the same error response that resource would return for missing or invalid input (see [RFC7285]).

Furthermore, it is RECOMMENDED that the server uses the following HTTP codes to indicate other errors, with the media type "application/alto-error+json".

* 429 (Too Many Requests): when the number of TIPS views open requests exceeds the server threshold. The server MAY indicate when to re-try the request in the "Re-Try After" headers.

It is RECOMMENDED that the server provide the ALTO/SSE support for the TIPS resource. Thus, the client can be notified of the version updates of all the TIPS views that it monitors and make better cross-resource transport decisions (see Section 8.2 for related considerations).

6.3. Open Example

6.3.1. Basic Example

For simplicity, assume that the ALTO server is using the Basic authentication. If a client with username "client1" and password "helloalto" wants to create a TIPS view of an ALTO Cost Map resource with resource ID "my-routingcost-map", it can send the request depicted in Figure 10.

```
POST /tips HTTP/1.1
Host: alto.example.com
Accept: application/alto-tips+json, application/alto-error+json
Authorization: Basic Y2xpZW50MTpoZWxsb2FsdG8K
Content-Type: application/alto-tipsparams+json
Content-Length: 41

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

Figure 10: Request Example of Opening a TIPS View

If the operation is successful, the ALTO server returns the message shown in Figure 11.

```
HTTP/1.1 200 OK
Content-Type: application/alto-tips+json
Content-Length: 255

{
  "tips-view-uri": "https://alto.example.com/tips/2718281828",
  "tips-view-summary": {
    "updates-graph-summary": {
      "start-seq": 101,
      "end-seq": 106,
      "start-edge-rec" : {
        "seq-i": 0,
        "seq-j": 105
      }
    }
  }
}
```

Figure 11: Response Example of Opening a TIPS View

6.3.2. Example using Digest Authentication

Below is another example of the same query using Digest authentication, a mandatory authentication method of ALTO servers as defined in Section 8.3.5 of [RFC7285]. The content of the response is the same as in Figure 11 and thus omitted for simplicity.

```
POST /tips HTTP/1.1
Host: alto.example.com
Accept: application/alto-tips+json, application/alto-error+json
Authorization: Basic Y2xpZW50MTpoZWxsb2FsdG8K
Content-Type: application/alto-tipsparams+json
Content-Length: 41

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

HTTP/1.1 401 UNAUTHORIZED
WWW-Authenticate: Digest
    realm="alto.example.com",
    qop="auth",
    algorithm="MD5",
    nonce="173b5aba4242409ee2ac3a4fd797f9d7",
    opaque="a237ff9ab865379a69d9993162ef55e4"

POST /tips HTTP/1.1
Host: alto.example.com
Accept: application/alto-tips+json, application/alto-error+json
Authorization: Digest
    username="client1",
    realm="alto.example.com",
    uri="/tips",
    qop=auth,
    algorithm=MD5,
    nonce="173b5aba4242409ee2ac3a4fd797f9d7",
    nc=00000001,
    cnonce="ZTg3MTI3NDFmMDQ0NzI1MDQ3MWE3ZTFjZmM5MTNiM2I=",
    response="8e937ae696c1512e4f990fa21c7f9347",
    opaque="a237ff9ab865379a69d9993162ef55e4"
Content-Type: application/alto-tipsparams+json
Content-Length: 41

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


HTTP/1.1 200 OK
Content-Type: application/alto-tips+json
Content-Length: 258

{....}
```

Figure 12: Open Example with Digest Authentication

6.3.3. Example using ALTO/SSE

This section gives an example of receiving incremental updates of the TIPS view summary using ALTO/SSE [RFC8895]. Consider the tips-sse resource, as announced by the IRD in Figure 6, which provides ALTO/SSE for the update-my-cost-tips resource, a client may send the following request to receive updates of the TIPS view (authentication is omitted for simplicity).

```
POST /updates/tips HTTP/1.1
Host: alto.example.com
Accept: text/event-stream,application/alto-error+json
Content-Type: application/alto-updatestreamparams+json
Content-Length: 76

{
  "add": {
    "tips-123": { "resource-id": "update-my-cost-tips" }
  }
}
```

Figure 13: Example of Monitoring TIPS view with ALTO/SSE

Then, the client will be able to receive the TIPS view summary as follows.

```
HTTP/1.1 200 OK
Connection: keep-alive
Content-Type: text/event-stream

event: application/alto-tips+json,tips-123
data: {
data:   "tips-view-uri": "https://alto.example.com/tips/2718281828",
data:   "tips-view-summary": {
data:     "updates-graph-summary": {
data:       "start-seq": 101,
data:       "end-seq": 106,
data:       "start-edge-rec" : {
data:         "seq-i": 0,
data:         "seq-j": 105
data:       }
data:     }
data:   }
data: }
```

When there is an update to the TIPS view, for example, the end-seq is increased by 1, the client will be able to receive the incremental update of the TIPS view summary as follows.

```
event: application/merge-patch+json,tips-123
data: {
data:   "tips-view-summary": {
data:     "updates-graph-summary": {
data:       "end-seq": 107
data:     }
data:   }
data: }
```

7. TIPS Data Transfers - Client Pull

TIPS allows an ALTO client to retrieve the content of an update item from the updates graph, with an update item defined as the content (incremental update or snapshot) on an edge in the updates graph.

7.1. Request

The client sends an HTTP GET request, where the media type of an update item resource MUST be the same as the "media-type" field of the update item on the specified edge in the updates graph.

The GET request MUST have the following format:

```
GET /<tips-view-path>/ug/<i>/<j>
HOST: <tips-view-host>
```

For example, consider the updates graph in Figure 4. If the client wants to query the content of the first update item (0 -> 101) whose media type is "application/alto-costmap+json", it sends a request to "/tips/2718281828/ug/0/101" and sets the "Accept" header to "application/alto-costmap+json,application/alto-error+json". See Section 7.3 for a concrete example.

7.2. Response

If the request is valid (ug/<i>/<j> exists), the response is encoded as a JSON object whose data format is indicated by the media type.

A client MAY conduct proactive fetching of future updates, by long polling updates that have not been provided in the directory yet. For such updates, the client MUST indicate all media types that may appear. It is RECOMMENDED that the server allows for at least the long polling of <end-seq> -> <end-seq + 1>.

Hence, the server processing logic MUST be:

* If ug/<i>/<j> exists: return content using encoding.

* Else if long polling ug/<i>/<j> is acceptable: put request in a backlog queue, then either a response is triggered when the content is ready or the request is interrupted, e.g., by a network error.

* Else: return error.

It is RECOMMENDED that the server uses the following HTTP codes to indicate errors, with the media type "application/alto-error+json", regarding update item requests.

* 404 (Not Found): if the requested update does not exist, or the requested TIPS view does not exist or is closed by the server.

* 410 (Gone): if an update has a seq that is smaller than the start-seq.

* 415 (Unsupported Media Type): if the media type(s) accepted by the client does not include the media type of the update chosen by the server.

* 425 (Too Early): if the seq exceeds the server long-polling window

* 429 (Too Many Requests): when the number of pending (long-poll) requests exceeds the server threshold. The server MAY indicate when to re-try the request in the "Re-Try After" headers.

7.3. Example

Assume the client wants to get the contents of the update item on edge 0 to 101. The format of the request is shown in Figure 14.

```
GET /tips/2718281828/ug/0/101 HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json, \
          application/alto-error+json
```

Figure 14: GET Example

The response is shown in Figure 15.

```
HTTP/1.1 200 OK
Content-Type: application/alto-costmap+json
Content-Length: 50

{ ... full replacement of my-routingcost-map ... }
```

Figure 15: Response to a GET Request

7.4. New Next Edge Recommendation

While intended TIPS usage is for the client to receive a recommended starting edge in the TIPS summary, consume that edge, then construct all future URIs by incrementing the sequence count by 1, there may be cases in which the client needs to request a new next edge to consume. For example, if a client has an open TIPS view yet has not polled in a while, the client may request the next logical incremental URI but the server has compacted the updates graph so it no longer exists. Thus, the client MAY request a new next edge to consume based on its current version of the resource.

7.4.1. Request

An ALTO client requests that the server provide a next edge recommendation for a given TIPS view by sending an HTTP POST request with the media type "application/alto-tipsparams+json". The URL of the request MUST have the format of

<tips-view-path>/ug

and the HOST field MUST be the <tips-view-host>.

The POST body has the same format as the TIPSReq Figure 7. The resource-id MUST be the same as the resource ID used to create the TIPS view, and the optional input field MUST NOT be present.

7.4.2. Response

The response to a valid request MUST be a JSON merge patch to the object of type AddTIPSResponse (defined in Section 6.2), denoted as media type "application/merge-patch+json". The "update-graph-summary" field MUST be present in the response and hence its parent field "tips-view-summary" MUST be present as well.

If the tag field is present in the request, the server MUST check if any version within the range [start-seq, end-seq] has the same tag value. If the version exists, e.g., denoted as tag-seq, the server MUST compute the paths from both tag-seq and 0 to the end-seq, and choose the one with the minimal cost. The cost MAY be implementation specific, e.g., number of messages, accumulated data size, etc. The first edge of the selected path MUST be returned as the recommended next edge.

If the tag field is NOT present, it MUST be interpreted as the tag-seq is 0.

It is RECOMMENDED that the server uses the following HTTP codes to indicate errors, with the media type "application/alto-error+json", regarding new next edge requests.

* 404 (Not Found): if the requested TIPS view does not exist or is closed by the server.

7.4.3. Example

We give an example of the new next edge recommendation service. Assume that a client already creates a TIPS view as in Section 6.3, whose updates graph is as shown in Figure 2. Now assume that the client already has tag 0881080 whose corresponding sequence number is 103, and sends the following new next edge recommendation request (authentication is omitted for simplicity):

```
POST /tips/2718281828/ug HTTP/1.1
HOST alto.example.com
Accept: application/merge-patch+json, application/alto-error+json
Content-Type: application/alto-tipsparams+json
Content-Length: 62

{
  "resource-id": "my-routingcost-map",
  "tag": "0881080"
}
```

According to Figure 2, there are 3 potential paths: 103 -> 104 -> 105 -> 106, 103 -> 105 -> 106, and 0 -> 105 -> 106. Assume that the server chooses shortest update path by the accumulated data size and the best path is 103 -> 105 -> 106. Thus, the server responds with the following message:

```
HTTP/1.1 200 OK
Content-Type: application/merge-patch+json
Content-Length: 193

{
  "tips-view-summary": {
    "updates-graph-summary": {
      "start-seq": 101,
      "end-seq": 106,
      "start-edge-rec": {
        "seq-i": 103,
        "seq-j": 105
      }
    }
  }
}
```

8. Operation and Processing Considerations

TIPS has some common operational considerations as ALTO/SSE [RFC8895], including:

* server choosing update messages (Section 9.1 of [RFC8895]);

* client processing update messages (Section 9.2 of [RFC8895]);

* updates of filtered map services (Section 9.3 of [RFC8895]);

* updates of ordinal mode costs (Section 9.4 of [RFC8895]).

There are also some operation considerations specific to TIPS, which we discuss below.

8.1. Considerations for Load Balancing

There are two levels of load balancing in TIPS. The first level is to balance the load of TIPS views for different clients, and the second is to balance the load of incremental updates.

Load balancing of TIPS views can be achieved either at the application layer or at the infrastructure layer. For example, an ALTO server MAY set <tips-view-host> to different subdomains to distribute TIPS views, or simply use the same host of the TIPS service and rely on load balancers to distribute the load.

TIPS allows a client to make concurrent pulls of incremental updates for the same TIPS view potentially through different HTTP connections. As a consequence, it introduces additional complexities

when the ALTO server is being load balanced. For example, a request may be directed to a wrong backend server and get incorrectly processed if the following two conditions both hold:

* the backend servers are stateful, i.e., the TIPS view is created and stored only on a single server;

* the ALTO server is using layer-4 load balancing, i.e., the requests are distributed based on the TCP 5-tuple.

Thus, additional considerations are required to enable correct load balancing for TIPS, including:

* Use a stateless architecture: One solution is to follow the stateless computing pattern: states about the TIPS view are not maintained by the backend servers but are stored in a distributed database. Thus, concurrent requests to the same TIPS view can be processed on arbitrary stateless backend servers, which all fetches data from the same database.

* Configure the load balancers properly: In case when the backend servers are stateful, the load balancers must be properly configured to guarantee that requests of the same TIPS view always arrive at the same server. For example, an operator or a provider of an ALTO server MAY configure layer-7 load balancers that distribute requests based on the tips-view-path component in the URI.

8.2. Considerations for Cross-Resource Dependency Scheduling

Dependent ALTO resources result in cross-resource dependencies in TIPS. Consider the following pair of resources, where my-cost-map (C) is dependent on my-network-map (N). The updates graph for each resource is shown, along with links in between the respective updates graphs to show dependency:

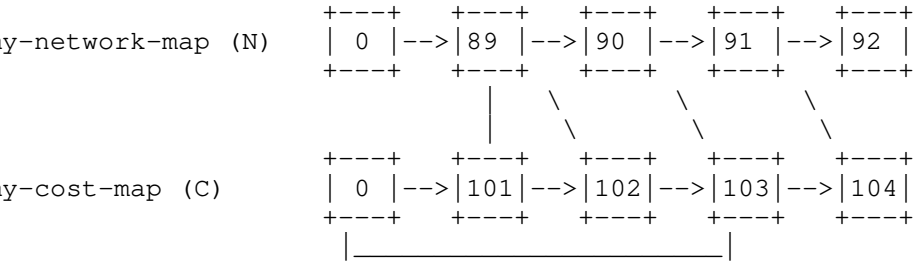


Figure 16: Example Dependency Model

In Figure 16, the cost-map versions 101 and 102 (denoted as C101 and C102) are dependent on the network-map version 89 (denoted as N89). The cost-map version 103 (C103) is dependent on the network-map version 90 (N90), and so on.

Thus, the client must decide the order in which to receive and apply the updates. The order may affect how fast the client can build a consistent view and how long the client needs to buffer the update.

* Example 1: The client requests N89, N90, N91, C101, C102 in that order. The client either gets no consistent view of the resources or has to buffer N90 and N91.

* Example 2: The client requests C101, C102, C103, N89. The client either gets no consistent view or has to buffer C103.

To get consistent ALTO information, a client must process the updates following the guidelines specified in Section 9.2 of [RFC8895]. If resource permits (i.e., sufficient updates can be buffered), an ALTO client can safely use long polling to fetch all the updates. This allows a client to build consistent views quickly as the updates are already stored in the buffer. Otherwise, it is RECOMMENDED to request

8.3. Considerations for Managing Shared TIPS Views

From a client's point of view, it sees only one copy of the TIPS view for any resource. However, on the server side, there are different implementation options, especially for common resources (e.g., network map or cost map) that may be frequently queried by many clients. Some potential options are listed below:

* An ALTO server creates one TIPS view of the common resource for each client.

* An ALTO server maintains one copy of the TIPS view for each common resource and all clients requesting the same resources use the same copy. There are two ways to manage the storage for the shared copy:

  - the ALTO server maintains the set of clients that have sent a polling request to the TIPS view, and only removes the view from the storage when the set becomes empty and no client immediately issues a new edge request;

  - the TIPS view is never removed from the storage.

Developers may choose different implementation options depending on criteria such as request frequency, available resources of the ALTO server, the ability to scale, and programming complexity.

8.4. Considerations for Offering Shortcut Incremental Updates

Besides the mandatory stepwise incremental updates (from i to i+1), an ALTO server MAY optionally offer shortcut incremental updates, or simple shortcuts, between two non-consecutive versions i and i+k (k > 1). Such shortcuts offer alternative paths in the update graph and can potentially speed up the transmission and processing of incremental updates, leading to faster synchronization of ALTO information, especially when the client has limited bandwidth and computation. However, implementors of an ALTO server must be aware that:

1. Optional shortcuts may increase the size of the update graph, in the worst case being the square of the number of updates (i.e., when a shortcut is offered for each version to all future versions).

2. Optional shortcuts require additional storage on the ALTO server.

3. Optional shortcuts may reduce concurrency when the updates do not overlap, e.g., when the updates apply to different parts of an ALTO resource. In such a case, the total size of the original updates is close to the size of the shortcut, but the original updates can be transmitted concurrently while the shortcut is transmitted in a single connection.

9. Security Considerations

The security considerations (Section 15 of [RFC7285]) of the base protocol fully apply to this extension. For example, the same authenticity and integrity considerations (Section 15.1 of [RFC7285]) still fully apply; the same considerations for the privacy of ALTO users (Section 15.4 of [RFC7285]) also still fully apply. Additionally, operators of the ALTO servers MUST follow the guidelines in [RFC9325] to avoid new TLS vulnerabilities discovered after [RFC7285] was published.

The additional services (addition of update read service and update push service) provided by this extension extend the attack surface described in Section 15.1.1 of [RFC7285]. The following sub-sections discuss the additional risks and their remedies.

9.1. TIPS: Denial-of-Service Attacks

Allowing TIPS views enables new classes of Denial-of-Service attacks. In particular, for the TIPS server, one or multiple malicious ALTO clients might create an excessive number of TIPS views, to exhaust the server resource and/or to block normal users from the accessing the service.

To avoid such attacks, the server MAY choose to limit the number of active views and reject new requests when that threshold is reached. TIPS allows predictive fetching and the server MAY also choose to limit the number of pending requests. If a new request exceeds the threshold, the server MAY log the event and return the HTTP status "429 Too many requests".

It is important to note that the preceding approaches are not the only possibilities. For example, it may be possible for TIPS to use somewhat more clever logic involving TIPS view eviction policies, IP reputation, rate-limiting, and compartmentalization of the overall threshold into smaller thresholds that apply to subsets of potential clients. If service availability is a concern, ALTO clients MAY establish service level agreements with the ALTO server.

9.2. ALTO Client: Update Overloading or Instability

The availability of continuous updates can also cause overload for an ALTO client, in particular, an ALTO client with limited processing capabilities. The current design does not include any flow control mechanisms for the client to reduce the update rates from the server. For example, TCP, HTTP/2 and QUIC provide stream and connection flow control data limits, which might help prevent the client from being overloaded. Under overloading, the client MAY choose to remove the information resources with high update rates.

Also, under overloading, the client may no longer be able to detect whether information is still fresh or has become stale. In such a case, the client should be careful in how it uses the information to avoid stability or efficiency issues.

10. IANA Considerations

IANA is requested to register the following media types from the registry available at [IANA-Media-Type]:

* application/alto-tips+json: as described in Section 6.2;

* application/alto-tipsparams+json: as described in Section 6.1;

Note to the RFC Editor: Please replace This-Document with the RFC number to be assigned to this document.

10.1. application/alto-tips+json Media Type

Type name: application

Subtype name: alto-tips+json

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type. See [RFC8259].

Security considerations: See the Security Considerations section of This-Document.

Interoperability considerations: N/A.

Published specification: Section 6.2 of This-Document.

Applications that use this media type: ALTO servers and ALTO clients either stand alone or are embedded within other applications.

Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: N/A

Magic number(s): N/A

File extension(s): This document uses the media type to refer to protocol messages and thus does not require a file extension.

Macintosh file type code(s): N/A

Person and email address to contact for further information: See Aut hors' Addresses section.

Intended usage: COMMON

Restrictions on usage: N/A

Author: See Authors' Addresses section.

Change controller: Internet Engineering Task Force (mailto:iesg@ietf.org).

Provisional registration?: No

10.2. application/alto-tipsparams+json Media Type

Type name: application

Subtype name: alto-tipsparams+json

Required parameters: N/A

Optional parameters: N/A

Encoding considerations: Encoding considerations are identical to those specified for the "application/json" media type. See [RFC8259].

Security considerations: See the Security Considerations section of This-Document.

Interoperability considerations: N/A.

Published specification: Section 6.1 of This-Document.

Applications that use this media type: ALTO servers and ALTO clients either stand alone or are embedded within other applications.

Fragment identifier considerations: N/A

Additional information:

Deprecated alias names for this type: N/A

Magic number(s): N/A

File extension(s): This document uses the media type to refer to protocol messages and thus does not require a file extension.

Macintosh file type code(s): N/A

Person and email address to contact for further information: See Aut hors' Addresses section.

Intended usage: COMMON

Restrictions on usage: N/A

Author: See Authors' Addresses section.

Change controller: Internet Engineering Task Force (mailto:iesg@ietf.org).

Provisional registration?: No

11. References

11.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/rfc/rfc3986>.

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <https://www.rfc-editor.org/rfc/rfc7285>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[RFC8895] Roome, W. and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Incremental Updates Using Server-Sent Events (SSE)", RFC 8895, DOI 10.17487/RFC8895, November 2020, <https://www.rfc-editor.org/rfc/rfc8895>.

[RFC9112] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP/1.1", STD 99, RFC 9112, DOI 10.17487/RFC9112, June 2022, <https://www.rfc-editor.org/rfc/rfc9112>.

[RFC9113] Thomson, M., Ed. and C. Benfield, Ed., "HTTP/2", RFC 9113, DOI 10.17487/RFC9113, June 2022, <https://www.rfc-editor.org/rfc/rfc9113>.

[RFC9114] Bishop, M., Ed., "HTTP/3", RFC 9114, DOI 10.17487/RFC9114, June 2022, <https://www.rfc-editor.org/rfc/rfc9114>.

[RFC9325] Sheffer, Y., Saint-Andre, P., and T. Fossati, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 9325, DOI 10.17487/RFC9325, November 2022, <https://www.rfc-editor.org/rfc/rfc9325>.

11.2. Informative References

[IANA-Media-Type] "Media Types", June 2023, <https://www.iana.org/assignments/media-types/media-types.xhtml>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/rfc/rfc4122>.

[RFC9205] Nottingham, M., "Building Protocols with HTTP", BCP 56, RFC 9205, DOI 10.17487/RFC9205, June 2022, <https://www.rfc-editor.org/rfc/rfc9205>.

Appendix A. A High-Level Deployment Model

Conceptually, the TIPS system consists of three types of resources:

* (R1) TIPS frontend to create TIPS views.

* (R2) TIPS view directory, which provides metadata (e.g., references) about the network resource data.

* (R3) The actual network resource data, encoded as complete ALTO network resources (e.g., cost map, network map) or incremental updates.

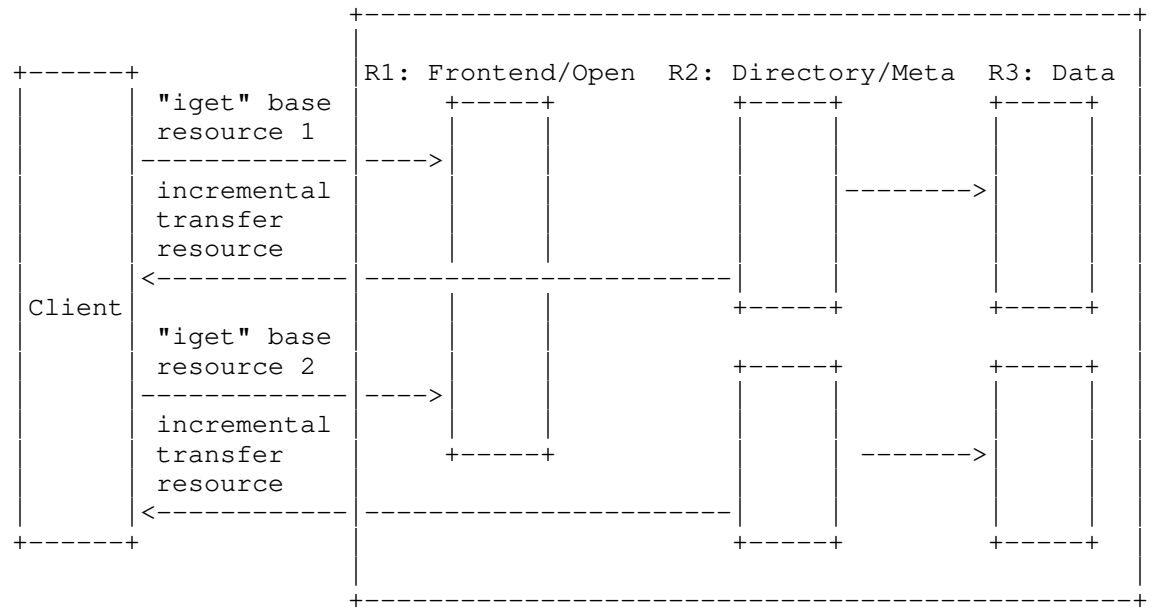


Figure 17: Sample TIPS Deployment Model

Design Point: Component Resource Location

* Design 1 (Single): all the three resource types at the same, single server (accessed via relative reference)

* Design 2 (Flexible): all three resource types can be at their own server (accessed via absolute reference)

* Design 3 (Dir + Data): R2 and R3 must remain together, though R1 might not be on the same server

This document supports Design 1 and Design 3. For Design 1, the TIPS service simply needs to always use the same host for the TIPS views. For Design 3, the TIPS service can set tips-view-host to a different server. Note that the deployment flexibility is at the logical level, as these services can be distinguished by different paths and potentially be routed to different physical servers by layer-7 load balancing. See Section 8.1 for a discussion on load balancing considerations. Future documents may extend the protocol to support Design 2.

Appendix B. Conformance to "Building Protocols with HTTP" Best Current Practices

This specification adheres fully to [RFC9205] as further elaborated below:

* TIPS does not "redefine, refine, or overlay the semantics of generic protocol elements such as methods, status codes, or existing header fields" and instead focuses on "protocol elements that are specific to [the TIPS] application -- namely, [its] HTTP resources" (Section 3.1 of [RFC9205]).

* There are no statically defined URI components (Section 3.2 of [RFC9205]).

* No minimum version of HTTP is specified by TIPS which is recommended (Section 4.1 of [RFC9205]).

* The TIPS design follows the advice that "When specifying examples of protocol interactions, applications should document both the request and response messages with complete header sections, preferably in HTTP/1.1 format" (Section 4.1 of [RFC9205]).

* TIPS uses URI templates which is recommended (Section 4.2 of [RFC9205]).

* TIPS follows the pattern that "a client will begin interacting with a given application server by requesting an initial document that contains information about that particular deployment, potentially including links to other relevant resources. Doing so ensures that the deployment is as flexible as possible (potentially spanning multiple servers), allows evolution, and also allows the application to tailor the "discovery document" to the client" (Section 4.4.1 of [RFC9205]).

* TIPS uses existing HTTP schemes (Section 4.4.2 of [RFC9205]).

* TIPS defines its errors "to use the most applicable status code" (Section 4.6 of [RFC9205]).

* TIPS does not "make assumptions about the relationship between separate requests on a single transport connection; doing so breaks many of the assumptions of HTTP as a stateless protocol and will cause problems in interoperability, security, operability, and evolution" (Section 4.11 of [RFC9205]). The only relationship between requests is that a client must first discover where a TIPS view of a resource will be served, which is consistent with the URI discovery in Section 4.4.1 of [RFC9205].

Appendix C. Push-mode TIPS using HTTP Server Push

TIPS allows ALTO clients to subscribe to incremental updates of an ALTO resource, and the specification in this document is based on the current best practice of building such a service using native HTTP. Earlier versions of this document had investigated the possibility of enabling push-mode TIPS, i.e., by taking advantage of the server push feature in HTTP/2 and HTTP/3.

In the ideal case, push-mode TIPS can potentially improve performance (e.g., latency) in more dynamic environments and use cases, with wait-free message delivery. Using native server push also results in minimal changes to the current protocol. While not adopted due to the lack of server push support and increased protocol complexity, push-mode TIPS remains a potential direction of protocol improvement.

Appendix D. Persistent HTTP Connections

Previous versions of this document use persistent HTTP connections to detect the liveness of clients. This design, however, does not conform well with the best current practice of HTTP. For example, if an ALTO client is accessing a TIPS view over an HTTP proxy, the connection is not established directly between the ALTO client and the ALTO server, but between the ALTO client and the proxy and between the proxy and the ALTO server. Thus, using persistent connections may not correctly detect the right liveness state.

Acknowledgments

The authors of this document would like to thank Mark Nottingham and Spencer Dawkins for providing invaluable reviews of earlier versions of this document, Adrian Farrel, Qin Wu, and Jordi Ros Giralt for their continuous feedback, Russ White, Donald Eastlake, Martin Thomson, Bernard Adoba, Spencer Dawkins, Linda Dunbar and Sheng Jiang for the directorate reviews, Martin Duke for the Area Director review, Francesca Palombini, Wesley Eddy, Roman Danyliw, Murray Kucherawy and Zaheduzzaman Sarker for the telechat and IESG reviews, and Mohamed Boucadair for shepherding the document.

Authors' Addresses

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

Roland Schott
Deutsche Telekom
Ida-Rhodes-Straße 2
64295 Darmstadt
Germany
Email: Roland.Schott@telekom.de

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

Lauren Delwiche
Yale University
51 Prospect Street
New Haven, 3408
United States of America
Email: lauren.delwiche@yale.edu

Lachlan Keller
Yale University
51 Prospect Street
New Haven, 3408
United States of America
Email: lachlan.keller@yale.edu

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: 13 January 2023

J. Zhang
Tongji University
D. Dhody
Huawei Technologies
K. Gao
Sichuan University
R. Schott
Deutsche Telekom

12 July 2022

A Yang Data Model for OAM and Management of ALTO Protocol
draft-ietf-alto-oam-yang-01

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/ietf-wg-alto/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 13 January 2023.

Table of Contents

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.

Prefix	YANG module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
key-chain	ietf-key-chain	[RFC8177]

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.

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

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.

4.4. Overview of ALTO O&M Data Model for Reference ALTO Architecture

Figure 1 shows a reference architecture for ALTO server implementation and YANG modules that server components implement. The server manager, information resource manager and data source listeners need to implement ietf-alto.yang (see Section 5). The performance monitor and logging and fault manager need to implement ietf-alto-stats.yang (see Section 6).

The data broker and algorithm plugins are not in the scope of the data model defined in this document. But user specified YANG modules can be applied to different algorithm plugins by augmenting the data model defined in this document (see Section 7).

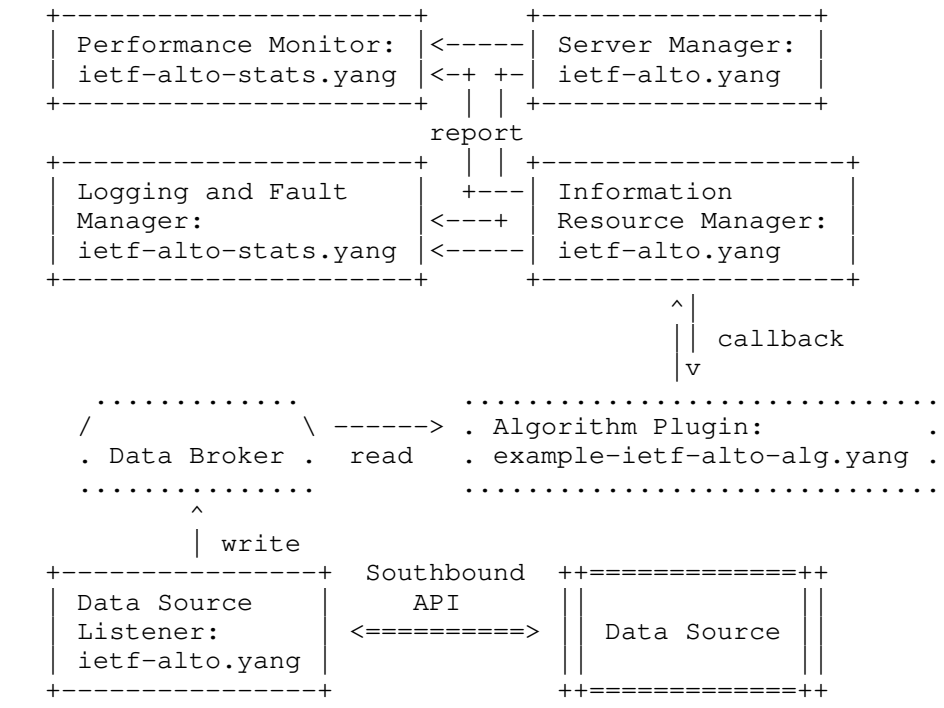


Figure 1: A Reference ALTO Server Architecture and YANG Modules

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 adminstrated 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 listen
     |  +---u alto-server-listen-stack-grouping
     +--rw cost-type* [cost-type-name]
     |  +--rw cost-type-name    string
     |  +--rw cost-mode         identityref
     |  +--rw cost-metric       identityref
     +--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
     |     |     +---u algorithm
     |     +--:(costmap)
     |     |  +--rw alto-costmap-params
     |     |     +--rw filtered?           boolean
```

```
|      |      +---u filt-costmap-cap
|      |      +---u algorithm
|      +--:(endpointcost)
|      |  +--rw alto-endpointcost-params
|      |     +---u filt-costmap-cap
|      |     +---u algorithm
|      +--:(endpointprop)
|      |  +--rw alto-endpointprop-params
|      |     +--rw prop-types*   string
|      |     +---u algorithm
|      +--:(propmap) {propmap}?
|      |  +--rw alto-propmap-params
|      |     +---u algorithm
|      +--:(cdni) {cdni}?
|      |  +--rw alto-cdni-params
|      |     +---u algorithm
|      +--:(update) {incr-update}?
|         +--rw alto-update-params
|            +---u 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)?
      +--:(yang-datastore)
      |  +--rw yang-datastore-source-params
      |     +--rw source-path
      |     |       yang:xpath1.0
      |     +--rw (restconf-endpoint)?
      |        +--:(local)
      |        +--:(remote)
      |           +--rw restconf-endpoint-params
      |              +---u rcc:restconf-client-listen-stack-grouping
      +--:(prometheus)
         +--rw prometheus-source-params
            +--rw source-uri    inet:uri
            +--rw query-data?   string
```

5.2. Meta Information of ALTO Server

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

The "listen" contains all the configurations for the whole server listen stack across HTTP layer, TLS layer and TCP layer.

```
  grouping alto-server-listen-stack-grouping
    +-- (transport)
       +--:(http) {http-listen}?
       |  +-- http
       |     +-- tcp-server-parameters
       |     |  +---u tcp:tcp-server-grouping
       |     +-- http-server-parameters
       |     |  +---u http:http-server-grouping
       |     +-- alto-server-parameters
       +--:(https)
          +-- https
             +-- tcp-server-parameters
                +---u tcp:tcp-server-grouping
                +-- tls-server-parameters
                |  +---u tls:tls-server-grouping
                +-- http-server-parameters
                |  +---u http:http-server-grouping
                +-- alto-server-parameters
```

TODO: A "base-uri" for ALTO clients to access may still be needed.

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.

```
module: ietf-alto
  +--rw alto-server
     +--rw listen
     |  +---u alto-server-listen-stack-grouping
     +--rw cost-type* [cost-type-name]
     |  +--rw cost-type-name    string
     |  +--rw cost-mode         identityref
     |  +--rw cost-metric       identityref
     +--rw meta* [meta-key]
     |  +--rw meta-key      string
     |  +--rw meta-value    string
     ...
```

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 resoruce entry provide 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 a 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 categories 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 altorithms 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]
     |  +--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
     |     |     +---u algorithm
     |     +--:(costmap)
     |     |  +--rw alto-costmap-params
     |     |     +--rw filtered?           boolean
     |     |     +---u filt-costmap-cap
     |     |     +---u algorithm
     |     +--:(endpointcost)
     |     |  +--rw alto-endpointcost-params
     |     |     +---u filt-costmap-cap
     |     |     +---u algorithm
     |     +--:(endpointprop)
     |     |  +--rw alto-endpointprop-params
     |     |     +--rw prop-types*   string
     |     |     +---u algorithm
```

```
        |     +--:(propmap) {propmap}?
        |     |  +--rw alto-propmap-params
        |     |     +---u algorithm
        |     +--:(cdni) {cdni}?
        |     |  +--rw alto-cdni-params
        |     |     +---u algorithm
        |     +--:(update) {incr-update}?
        |        +--rw alto-update-params
        |           +---u algorithm
        |
       ...

     grouping filt-costmap-cap
       +-- cost-type-names*            string
       +-- cost-constraints?           boolean
       +-- max-cost-types?             uint32 {multi-cost}?
       +-- testable-cost-type-names*   string {multi-cost}?
       +-- calendar-attributes {cost-calendar}?
          +-- cost-type-names*       string
          +-- time-interval-size     decimal64
          +-- number-of-intervals    uint32
```

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
           |     +--rw (restconf-endpoint)?
           |        +--:(local)
           |        +--:(remote)
           |           +--rw restconf-endpoint-params
           |              +---u rcc:restconf-client-listen-stack-grouping
           +--:(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 from a YANG model-driven data store.

It supports two types of endpoints: local and remote.

* For a local endpoint, the YANG data is located the data from the same YANG model-driven data store supplying the current ALTO O&M data model. Therefore, the ALTO data source listener retrieves the data using the internal API provided by the data store.

* For a remote endpoint, the ALTO data source listener establishes an HTTP connection to the remote RESTCONF server, and retrieve the data using the RESTCONF API.

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-uir is used to establish the connection with the Prometheus server. The query-data is used to speficify the potential query expression in PromQL.

5.5. Model for ALTO Server-to-server Communication

In practice, multiple ALTO servers can be deployed for scalability. That may require communication among different ALTO servers.

The YANG module defined in this document contains the configuration for the communication between two ALTO servers.

TODO: this is still under the open discussion status.

6. Design of ALTO O&M Statistics Data Model

6.1. Model for ALTO Logging and Fault Management

As section 16.2.1 and section 16.2.3 of [RFC7285] suggest, the YANG data module defined in this document contains statistics for logging and failure detection.

NOTE: The detailed YANG module will appear in the future version.

6.2. Model for ALTO-specific Performance Monitoring

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:
    +--ro num-total-req?         yang:counter32
    +--ro num-total-succ?        yang:counter32
    +--ro num-total-fail?        yang:counter32
    +--ro num-total-last-req?    yang:counter32
    +--ro num-total-last-succ?   yang:counter32
    +--ro num-total-last-fail?   yang:counter32
  augment /alto:alto-server/alto:resource:
    +--ro num-res-upd?    yang:counter32
    +--ro res-mem-size?   yang:counter32
    +--ro res-enc-size?   yang:counter32
    +--ro num-res-req?    yang:counter32
    +--ro num-res-succ?   yang:counter32
    +--ro num-res-fail?   yang:counter32
  augment /alto:alto-server/alto:resource/alto:resource-params
            /alto:networkmap/alto:alto-networkmap-params:
    +--ro num-map-pid?   yang:counter32
  augment /alto:alto-server/alto:resource/alto:resource-params
            /alto:propmap/alto:alto-propmap-params:
    +--ro num-map-entry?   yang:counter32
  augment /alto:alto-server/alto:resource/alto:resource-params
            /alto:cdni/alto:alto-cdni-params:
    +--ro num-base-obj?   yang:counter32
  augment /alto:alto-server/alto:resource/alto:resource-params
            /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

  augment /alto:alto-server/alto:resource/alto:resource-params
            /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 internel 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@2022-07-11.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";
  }

  import ietf-tcp-server {
    prefix tcp;
    reference
      "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
  }

  import ietf-tls-server {
    prefix tls;
    reference
      "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
  }

  import ietf-http-server {
    prefix http;
    reference
      "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
  }

  import ietf-restconf-client {
    prefix rcc;
    reference
      "RFC IIII: YANG Groupings for RESTCONF Clients and RESTCONF
       Servers";
  }

  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
       (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.

       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 "2022-07-11" {
      description
        "Initial Version.";
      reference
        "RFC XXXX: A YANG Data Model for OAM and Management of ALTO
         Protocol.";
    }

    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.";
    }

    // Base identities

    identity resource-type {
```

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

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

  identity cost-mode {
    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.";
  }

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

  // Identities for ALTO information resources

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

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

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

  // Identities for predefined data sources
```

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

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

// Identities for cost mode

identity numerical {
  base cost-mode;
  description
    "This mode indicates that it is safe to perform numerical
     operations";
}

identity ordinal {
  base cost-mode;
  description
    "This mode indicates that the cost values in a cost map
     represent ranking";
}

identity array {
  if-feature "path-vector";
  base cost-mode;
  description
    "This mode indicates that every cost value in the response body
     of a (Filtered) Cost Map or an Endpoint Cost Service MUST be
     interpreted as a JSON array.";
}

// Identities for cost metrics

identity routingcost {
  base cost-metric;
  description
    "This metric conveys a generic measure for the cost of routing
     traffic from a source to a destination.";
}

identity ane-path {
  if-feature "path-vector";
```

```
      base cost-metric;
      description
        "This metric indicates that the value of such a cost type
         conveys an array of Abstract Network Element (ANE) names,
         where each ANE name uniquely represents an ANE traversed by
         traffic from a source to a destination.";
    }

    identity delay-ow {
      if-feature "performance-metrics";
      base cost-metric;
      description
        "Section 4.1 of RFC XXXX";
    }

    identity delay-rt {
      if-feature "performance-metrics";
      base cost-metric;
      description
        "Section 4.2 of RFC XXXX";
    }

    identity delay-variation {
      if-feature "performance-metrics";
      base cost-metric;
      description
        "Section 4.3 of RFC XXXX";
    }

    identity lossrate {
      if-feature "performance-metrics";
      base cost-metric;
      description
        "Section 4.4 of RFC XXXX";
    }

    identity hopcount {
      if-feature "performance-metrics";
      base cost-metric;
      description
        "Section 4.5 of RFC XXXX";
    }

    identity tput {
      if-feature "performance-metrics";
      base cost-metric;
      description
        "Section 5.1 of RFC XXXX";
```

```
  }

  identity bw-residual {
    if-feature "performance-metrics";
    base cost-metric;
    description
      "Section 5.2 of RFC XXXX";
  }

  identity bw-available {
    if-feature "performance-metrics";
    base cost-metric;
    description
      "Section 5.3 of RFC XXXX";
  }

  // Features

  feature http-listen {
    description
      "The 'http-listen' feature is only used for test depolyment.
       According to Sec 8.3.5 of RFC 7285, it shouldn't be used in
       the production depolyment.";
  }

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

  feature incr-update {
    description
      "Support incremental update extension.";
    reference
      "RFC 8895: Application-Layer Traffic Optimization (ALTO)
       Incremental Updates Using Server-Sent Events (SSE)";
  }

  feature cost-calendar {
    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 9240: An ALTO Extension: Entity Property Maps";
  }

  feature cdni {
    description
      "Support CDNi extension.";
    reference
      "RFC 9241: Content Delivery Network Interconnection (CDNI)
       Request Routing: CDNI Footprint and Capabilities
       Advertisement using ALTO";
  }

  feature path-vector {
    description
      "Support path vector extension.";
    reference
      "RFC XXXX: An ALTO Extension: Path Vector";
  }

  feature performance-metrics {
    description
      "Support performance metrics extension.";
    reference
      "RFC XXXX: ALTO Performance Cost Metrics";
  }

  // Groupings

  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.";
    }

    grouping alto-server-grouping {
      description
        "A reuseable grouping for configuring an ALTO server without
         any consideration for how underlying transport sessions are
         established.";
      // TODO: ALTO specific server-level configuration
    }

    grouping alto-server-listen-stack-grouping {
      description
        "A reuseable grouping for configuring an ALTO server
         'listen' protocol stack for a single connection.";
      choice transport {
        mandatory true;
        description
          "Selects between available transports.";
        case http {
          if-feature "http-listen";
          container http {
            description
              "Configures ALTO server stack assuming that
               TLS-termination is handled externally.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP server parameters
                 to avoid name collisions.";
              uses tcp:tcp-server-grouping {
```

```
                refine "local-port" {
                  default "80";
                  description
                    "The RESTCONF server will listen on the IANA-
                     assigned well-known port value for 'http'
                     (80) if no value is specified.";
                }
              }
            }
            container http-server-parameters {
              description
                "A wrapper around the HTTP server parameters
                 to avoid name collisions.";
              uses http:http-server-grouping;
            }
            container alto-server-parameters {
              description
                "A wrapper around the ALTO server parameters
                 to avoid name collisiions.";
              uses alto-server-grouping;
            }
          }
        }
        case https {
          container https {
            description
              "Configures ALTO server stack assuming that
               TLS-termination is handled internally.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP server parameters
                 to avoid name collisions.";
              uses tcp:tcp-server-grouping {
                refine "local-port" {
                  default "443";
                  description
                    "The ALTO server will listen on the IANA-
                     assigned well-known port value for 'https'
                     (443) if no value is specified.";
                }
              }
              container tls-server-parameters {
                description
                  "A wrapper around the TLS server parameters
                   to avoid name collisions.";
                uses tls:tls-server-grouping;
              }
              container http-server-parameters {
```

```
                description
                  "A wrapper around the HTTP server parameters
                   to avoid name collisions.";
                uses http:http-server-grouping;
              }
              container alto-server-parameters {
                description
                  "A wrapper around the ALTO server parameters
                   to avoid name collisions.";
                uses alto-server-grouping;
              }
            }
          }
        }
      }
    }

    // Top-level container

    container alto-server {
      description
        "The ALTO server instance.";
      container listen {
        description
          "Configure the ALTO server to listen for ALTO clients.";
        uses alto-server-listen-stack-grouping;
      }
      list cost-type {
        key "cost-type-name";
        leaf cost-type-name {
          type string;
          description
            "The name to reference cost type";
        }
        leaf cost-mode {
          type identityref {
            base cost-mode;
          }
          mandatory true;
          description
            "The referenced cost mode";
        }
        leaf cost-metric {
          type identityref {
            base 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-type;
          }
          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-type;
        }
        mandatory true;
        description
```

```
            "Source-type to be defined";
          }
          choice update-policy {
            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";
                choice restconf-endpoint {
                  case local {
                    // Use local API to access YANG datastore
                  }
                  case remote {
                    container restconf-endpoint-params {
                      uses rcc:restconf-client-listen-stack-grouping;
                    }
                  }
                }
              }
            }
            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";
            }
          }
          description
            "Data source specific configuration";
        }
        description
          "List of subscribed data sources.";
      }
    }
  }
  <CODE ENDS>
```

8.2. The ietf-alto-stats Module

```
  <CODE BEGINS> file "ietf-alto-stats@2022-07-11.yang"
  module ietf-alto-stats {
    yang-version 1.1;
    namespace
      "urn:ietf:params:xml:ns:yang:ietf-alto-stats";
    prefix "alto-stats";

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

    import ietf-alto {
      prefix alto;
      reference
        "RFC XXXX: A YANG Data Model for OAM and Management of ALTO
         Protocol.";
    }

    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
       (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 "2022-07-11" {
      description
        "Initial Version.";
      reference
        "RFC XXXX: A YANG Data Model for Operations, Administration,
         and Maintenance of ALTO Protocol.";
    }

    augment "/alto:alto-server" {
      description
        "Top-level statistics for the whole ALTO server.";
      leaf num-total-req {
        type yang:counter32;
        config false;
        description
          "The total number of ALTO requests received by this ALTO
           server.";
      }
      leaf num-total-succ {
        type yang:counter32;
        config false;
        description
          "The total number of successful responses sent by this ALTO
           server.";
```

```
      }
      leaf num-total-fail {
        type yang:counter32;
        config false;
        description
          "The total number of failed responses sent by this ALTO
           server.";
      }
      leaf num-total-last-req {
        type yang:counter32;
        config false;
        description
          "The total number of ALTO requests received within the last
           5 minutes.";
      }
      leaf num-total-last-succ {
        type yang:counter32;
        config false;
        description
          "The total number of successful responses sent by this ALTO
           server within the last 5 minutes.";
      }
      leaf num-total-last-fail {
        type yang:counter32;
        config false;
        description
          "The total number of failed responses sent by this ALTO
           server within the last 5 minutes.";
      }
    }

    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.";
      }
      leaf num-res-req {
        type yang:counter32;
        config false;
        description
          "The number of ALTO requests to this information resource.";
      }
      leaf num-res-succ {
        type yang:counter32;
        config false;
        description
          "The number of successful responses for requests to this
           information resource.";
      }
      leaf num-res-fail {
        type yang:counter32;
        config false;
        description
          "The total number of failed responses for requests to this
           information resource.";
      }
    }

    augment "/alto:alto-server/alto:resource/alto:resource-params"
          + "/alto:networkmap/alto:alto-networkmap-params" {
      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
          "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.";
      }
    }
  }
  <CODE ENDS>
```

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.

```
  URI: urn:ietf:params:xml:ns:yang:ietf-alto
  Registrant Contact: The IESG.
  XML: N/A; the requested URI is an XML namespace.

  URI: urn:ietf:params:xml:ns:yang:ietf-alto-stats
  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
  Namespace: urn:ietf:params:xml:ns:yang:ietf-alto
  Prefix: alto
  Reference: [RFCthis]

  Name: ietf-alto-stats
  Namespace: urn:ietf:params:xml:ns:yang: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

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/rfc/rfc3688>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/rfc/rfc6020>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/rfc/rfc6991>.

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <https://www.rfc-editor.org/rfc/rfc7285>.

[RFC7286] Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and H. Song, "Application-Layer Traffic Optimization (ALTO) Server Discovery", RFC 7286, DOI 10.17487/RFC7286, November 2014, <https://www.rfc-editor.org/rfc/rfc7286>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8177] Lindem, A., Ed., Qu, Y., Yeung, D., Chen, I., and J. Zhang, "YANG Data Model for Key Chains", RFC 8177, DOI 10.17487/RFC8177, June 2017, <https://www.rfc-editor.org/rfc/rfc8177>.

[RFC8189] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost Application-Layer Traffic Optimization (ALTO)", RFC 8189, DOI 10.17487/RFC8189, October 2017, <https://www.rfc-editor.org/rfc/rfc8189>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/rfc/rfc8340>.

[RFC8686] Kiesel, S. and M. Stiemerling, "Application-Layer Traffic Optimization (ALTO) Cross-Domain Server Discovery", RFC 8686, DOI 10.17487/RFC8686, February 2020, <https://www.rfc-editor.org/rfc/rfc8686>.

[RFC8895] Roome, W. and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Incremental Updates Using Server-Sent Events (SSE)", RFC 8895, DOI 10.17487/RFC8895, November 2020, <https://www.rfc-editor.org/rfc/rfc8895>.

[RFC8896] Randriamasy, S., Yang, R., Wu, Q., Deng, L., and N. Schwan, "Application-Layer Traffic Optimization (ALTO) Cost Calendar", RFC 8896, DOI 10.17487/RFC8896, November 2020, <https://www.rfc-editor.org/rfc/rfc8896>.

11.2. Informative References

[I-D.ietf-alto-cdni-request-routing-alto]
Seedorf, J., Yang, Y. R., Ma, K. J., Peterson, J., and J. J. Zhang, "Content Delivery Network Interconnection (CDNI) Request Routing: CDNI Footprint and Capabilities Advertisement using ALTO", Work in Progress, Internet-Draft, draft-ietf-alto-cdni-request-routing-alto-22, 16 February 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-alto-cdni-request-routing-alto-22>.

[I-D.ietf-alto-path-vector]
Gao, K., Lee, Y., Randriamasy, S., Yang, Y. R., and J. J. Zhang, "An ALTO Extension: Path Vector", Work in Progress, Internet-Draft, draft-ietf-alto-path-vector-25, 20 March 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-alto-path-vector-25>.

[I-D.ietf-alto-performance-metrics]
Wu, Q., Yang, Y. R., Lee, Y., Dhody, D., Randriamasy, S., and L. M. C. Murillo, "ALTO Performance Cost Metrics", Work in Progress, Internet-Draft, draft-ietf-alto-performance-metrics-28, 21 March 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-alto-performance-metrics-28>.

[I-D.ietf-alto-unified-props-new]
Roome, W., Randriamasy, S., Yang, Y. R., Zhang, J. J., and K. Gao, "An ALTO Extension: Entity Property Maps", Work in Progress, Internet-Draft, draft-ietf-alto-unified-props-new-24, 28 February 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-alto-unified-props-new-24>.

[RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <https://www.rfc-editor.org/rfc/rfc7921>.

[RFC7971] Stiemerling, M., Kiesel, S., Scharf, M., Seidel, H., and S. Previdi, "Application-Layer Traffic Optimization (ALTO) Deployment Considerations", RFC 7971, DOI 10.17487/RFC7971, October 2016, <https://www.rfc-editor.org/rfc/rfc7971>.

[RFC8346] Clemm, A., Medved, J., Varga, R., Liu, X., Ananthakrishnan, H., and N. Bahadur, "A YANG Data Model for Layer 3 Topologies", RFC 8346, DOI 10.17487/RFC8346, March 2018, <https://www.rfc-editor.org/rfc/rfc8346>.

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";
     }

     augment "/alto:alto-server/alto:resource/alto:resource-params"
           + "/alto:networkmap/alto:alto-networkmap-params"
           + "/alto:algorithm" {
       case l3-unicast-cluster {
         container l3-unicast-cluster-algorithm {
           leaf l3-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

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: 22 July 2024

J. Zhang
Tongji University
D. Dhody
Huawei Technologies
K. Gao
Sichuan University
R. Schott
Deutsche Telekom
Q. Ma
Huawei

19 January 2024

YANG Data Models for the Application-Layer Traffic Optimization (ALTO) Protocol
draft-ietf-alto-oam-yang-17

Abstract

This document defines a YANG data model for Operations, Administration, and Maintenance (OAM) & Management of the Application-Layer Traffic Optimization (ALTO) Protocol. The operator of an ALTO server can use this data model to (1) set up the ALTO server, (2) configure server discovery, (3) create, update and remove ALTO information resources, (4) manage the access control of each ALTO information resource, and (5) collect statistical data from the ALTO server. The application provider can also use this data model to configure ALTO clients to communicate with known ALTO servers.

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/ietf-wg-alto/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.

This Internet-Draft will expire on 22 July 2024.

Table of Contents

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 of an ALTO server can use this data model to:

* set up the ALTO server,

* configure server discovery,

* create, update and remove ALTO information resources,

* manage the access control of each ALTO information resource,

* collect statistical data of the ALTO server.

The application provider can also use this data model to configure ALTO clients to communicate with known ALTO servers.

Section 4.1 describes what is and is not in scope. Section 4.2 and Section 4.3 define more concrete requirements for the data model.

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: [RFC8189], [RFC8895], [RFC8896], [RFC9240], [RFC9241], [RFC9274], [RFC9275], and [RFC9439].

The detailed design of the data model is illustrated in Section 5 and Section 6. Some examples of how to extend this data model for specific ALTO server implementations are shown in Appendix A.

The YANG data models in this document conform to the Network Management Datastore Architecture (NMDA) defined in [RFC8342].

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. Acronyms and Abbreviations

This document uses the following acronyms:

ALTO: Application-Layer Traffic Optimization

CPU: Central Processing Unit

DNS: Domain Name System

HTTP: Hypertext Transfer Protocol

IRR: Internet Routing Registry

OAM: Operations, Administration, and Maintenance (Section 3 of [RFC6291])

O&M: OAM and Management (Section 3 of [RFC6291])

OAuth: Open Authorization

PID: Provider-defined Identifier in ALTO

TCP: Transmission Control Protocol

TLS: Transport Layer Security

URI: Uniform Resource Identifier

3.1. Tree Diagrams

The meaning of the symbols in the tree diagrams is defined in [RFC8340].

3.2. Prefixes in Data Node Names

The complete name of a data node or data model object includes a prefix, which indicates the YANG module in which the name is defined. In this document, the prefix is omitted when the YANG module is clear from the context; otherwise, the prefix is included. The prefixes indicating the corresponding YANG modules are shown in Table 1.

Prefix	YANG module	Reference
yang	ietf-yang-types	[RFC6991]
inet	ietf-inet-types	[RFC6991]
ds	ietf-datastores	[RFC8342]
yp	ietf-yang-push	[RFC8641]
ts	ietf-truststore	[I-D.ietf-netconf-trust-anchors]
tcp	ietf-tcp-server	[I-D.ietf-netconf-tcp-client-server]
tls	ietf-tls-server	[I-D.ietf-netconf-tls-client-server]
http	ietf-http-server	[I-D.ietf-netconf-http-client-server]
ncc	ietf-netconf-client	[I-D.ietf-netconf-netconf-client-server]
rcc	ietf-restconf-client	[I-D.ietf-netconf-restconf-client-server]

Table 1: Prefixes and corresponding YANG modules

3.3. Placeholders in Reference Statements

Note to the RFC Editor: This section is to be removed prior to publication.

This document contains placeholder values that need to be replaced with finalized values at the time of publication. This note summarizes all of the substitutions that are needed. No other RFC Editor instructions are specified elsewhere in this document.

Please apply the following replacements:

* BBBB --> the assigned RFC number for [I-D.ietf-netconf-trust-anchors]

* DDDD --> the assigned RFC number for [I-D.ietf-netconf-tcp-client-server]

* FFFF --> the assigned RFC number for [I-D.ietf-netconf-tls-client-server]

* GGGG --> the assigned RFC number for [I-D.ietf-netconf-http-client-server]

* HHHH --> the assigned RFC number for [I-D.ietf-netconf-netconf-client-server]

* IIII --> the assigned RFC number for [I-D.ietf-netconf-restconf-client-server]

* XXXX --> the assigned RFC number for this draft

4. Design Scope and Requirements

4.1. Scope of Data Models for ALTO O&M

The following items are in the scope of the data models specified in this document:

* Deploying an ALTO server/client.

* Operating and managing an ALTO server/client.

* Configuring functionality/capability configuration of ALTO services.

* Monitoring ALTO-related performance metrics.

This document does not normatively define any data model related to a 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])

Likewise, the specification does not cover considerations that are specific to the ALTO Transport Information Publication Service (TIPS) [I-D.ietf-alto-new-transport]. Specifically, considerations related to HTTP/3 are out of scope.

For convenience, examples of how related extensions can be defined are provided in the Appendices.

4.2. Basic Requirements

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

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

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 models 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 is designed to fit all the requirements listed in Section 4.

As shown in Figure 1, the top-level container 'alto' in the "ietf-alto" module contains a single 'alto-server' and a list 'alto-client'.

'alto-client' defines a list of configurations for other applications to bootstrap an ALTO client. These data nodes can also be used by data sources and information resource creation algorithms that are configured by an ALTO server instance.

The container 'alto-server' contains both configuration and operational data of an administrated ALTO server instance.

```
module: ietf-alto
  +--rw alto!
     +--rw alto-client* [client-id] {alto-client}?
     |  ...
     +--rw alto-server {alto-server}?
        +...
        +--rw auth-client* [client-id]
        |  ...
        +--rw role* [role-name]
        |  +--rw role-name    role-name
        |  +--rw client*      client-ref
        +--rw data-source* [source-id]
        |  ...
        +--rw resource* [resource-id]
           ...
```

Figure 1: IETF ALTO Tree Structure

5.2. Data Model for ALTO Client Operation and Management

As shown in Figure 2, the 'alto-client' contains a list of client-side configurations. Each 'alto-client' entry contains the following data nodes:

'client-id': A unique identifier that can be referenced by other applications.

'server-discovery-client': A container that is used to configure how this ALTO client discovers an ALTO server.

```
module: ietf-alto
  +--rw alto!
     +--rw alto-client* [client-id] {alto-client}?
     |  +--rw client-id                  string
     |  +--rw server-discovery-client
     |     +---u alto-server-discovery-client
     ...
```

Figure 2: IETF ALTO Client Subtree Structure

5.3. Data Model for Server-level Operation and Management

The ALTO server instance contains a set of data nodes for server-level operation and management for ALTO that are shown in Figure 3. This structure satisfies R1 - R4 in Section 4.2.

```
module: ietf-alto
  +--rw alto!
     ...
     +--rw alto-server {alto-server}?
        +--rw listen
        |  +---u alto-server-listen-stack
        +--rw server-discovery
        |  +---u alto-server-discovery
        +--rw logging-system
        |  +---u alto-logging-system
        +--rw cost-type* [cost-type-name]
        |  +--rw cost-type-name    cost-type-name
        |  +--rw cost-mode         identityref
        |  +--rw cost-metric       identityref
        |  +--rw description?      string
        |  +--rw cost-context {performance-metrics}?
        |     +--rw cost-source    identityref
        |     +--rw parameters
        |        +--rw (parameters)?
        +--rw meta* [meta-key]
        |  +--rw meta-key      meta-key
        |  +--rw meta-value    binary
        ...
```

Figure 3: IETF ALTO Server Level Subtree Structure

5.3.1. Data Model for ALTO Server Setup

To satisfy R1 in Section 4.2, the ALTO server instance contains the basic data nodes for the server setup that are detailed in the following subsections.

5.3.1.1. ALTO Server Listen Stack

The container 'listen' contains all the data nodes for the whole server listen stack across TCP, TLS, HTTP and application layers (Figure 4).

```
grouping alto-server:
  +-- base-uri?   inet:uri
grouping alto-server-listen-stack:
  +-- (transport)
     +--:(http) {http-listen}?
     |  +-- http
     |     +-- tcp-server-parameters
     |     |  +---u tcp:tcp-server-grouping
     |     +-- http-server-parameters
     |     |  +---u http:http-server-grouping
     |     +-- alto-server-parameters
     |        +---u alto-server
     +--:(https)
        +-- https
           +-- tcp-server-parameters
           |  +---u tcp:tcp-server-grouping
           +-- tls-server-parameters
           |  +---u tls:tls-server-grouping
           +-- http-server-parameters
           |  +---u http:http-server-grouping
           +-- alto-server-parameters
              +---u alto-server
```

Figure 4: IETF ALTO Server Groupings Structure

The 'transport' choice node enables which protocol layers to be configured. By default, two cases are defined for different deployment scenarios:

* The 'http' case is provided to support scenarios where the TLS-termination is handled by other external components, e.g., reverse proxies or ingress controllers.

* The 'https' case is provided to support scenarios where the whole HTTPS server listen stack including TLS is handled by the ALTO server itself.

5.3.1.2. ALTO Server Discovery Setup

In practice, for a large-scale network consisting of multiple administrative domains, the information about the network may be partitioned and distributed over multiple ALTO servers. That may require discovery and communication among different ALTO servers.

The "ietf-alto" module provides the configuration for how an ALTO server can be discovered by another ALTO server or client on demand (Figure 5). However, it does not contain any configuration for the communication among ALTO servers because the related solution has not become a standard. Future documents may extend it to fully support multi-domain scenarios.

```
grouping alto-server-discovery:
  +-- (server-discovery-manner)?
     +--:(reverse-dns) {xdom-disc}?
        +-- rdns-naptr-records
           +-- static-prefix*           inet:ip-prefix
           +-- dynamic-prefix-source*   data-source-ref
```

Figure 5: IETF ALTO Server Discovery Grouping Structure

The 'server-discovery' node provides configuration for the discovery of ALTO servers using a variety of mechanisms. The initial version of the "ietf-alto" module only defines the 'reverse-dns' case that is used to configure DNS NAPTR records for ALTO server discovery as suggested by [RFC7286] and [RFC8686]. It configures a set of endpoints that can be served by this ALTO server. The node contains two leaf lists. The 'static' list contains a list of manually configured endpoints. The 'dynamic' list points to a list of data sources to retrieve the endpoints dynamically. As suggested by [RFC7286] and [RFC8686], the IP prefixes of the endpoints configured by both 'static' and 'dynamic' lists will be translated into DNS NAPTR resource records for server discovery. The 'server-discovery-manner' choice can be augmented by the future modules to support other mechanisms.

5.3.2. Data Model for Logging Management

To satisfy R2 in Section 4.2, the ALTO server instance contains the the logging data nodes shown in Figure 6.

The 'logging-system' data node provides configuration to select a logging system to capture log messages generated by an ALTO server.

By default, 'syslog' is the only supported logging system. When selecting 'syslog', the related configuration is delegated to the configuration file of the syslog [RFC5424] server.

```
grouping alto-logging-system:
  +-- (logging-system)?
     +--:(syslog)
        +-- syslog-params
           +-- config-file?   inet:uri
```

Figure 6: IETF ALTO Logging System Grouping Structure

A specific server implementation can extend the 'logging-system' node to add other logging systems.

5.3.3. Data Model for ALTO-related Management

To satisfy R3 in Section 4.2, the data model contains the following ALTO-related management information (Figure 3):

* 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 is populated into the meta field of the default Information Resource Directory (IRD).

5.3.4. Data Model for Security Management

To satisfy R4 in Section 4.2, the data model leverages HTTP and TLS to provide basic security management for an ALTO server. All the related configurations are covered by the server listen stack.

5.4. Data Model for ALTO Server Configuration Management

5.4.1. Data Source Configuration Management

To satisfy R5-1 in Section 4.2, the ALTO server instance contains a list of 'data-source' entries to subscribe the data sources from which ALTO information resources are derived (Section 16.2.4 of [RFC7285]).

```
module: ietf-alto
  +--rw alto!
     ...
     +--rw alto-server {alto-server}?
        ...
        +--rw data-source* [source-id]
        |  +--rw source-id                    source-id
        |  +--rw source-type                  identityref
        |  +--rw source-params
        ...
```

Figure 7: IETF ALTO Server Data Source Subtree Structure

As shown in Figure 7, a 'data-source' list entry includes:

* A unique 'source-id' for resource creation algorithms to reference.

* The 'source-type' attribute to declare the type of the data source.

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

The 'data-source/source-params' node can be augmented for different types of data sources. Note that the purpose of this node is not to fully set up the communication mechanisms for specific data sources, but to maintain how data sources are configured and expose them to the ALTO server.

This data model only includes a basic structure for an ALTO server to correctly interact with a data source. The implementation-specific parameters of any certain data source can be augmented in another module. An example is included in Appendix A.3.

5.4.2. ALTO Information Resources Configuration Management

To satisfy R5-2 and R-3 in Section 4.2, the ALTO server instance contains a list of 'resource' entries (Figure 8). Each 'resource' entry contains the data nodes 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 resources.

Each 'resource' entry provides data nodes defining how to create or update an ALTO information resource. Adding a new 'resource' entry notifies the ALTO server to create a new ALTO information resource. Updating an existing 'resource' entry notifies the ALTO server to update the generation parameters (e.g., capabilities and the creation

algorithm) of an existing ALTO information resource. Removing an existing 'resource' entry will remove the corresponding ALTO information resource.

```
  module: ietf-alto
    +--rw alto!
       ...
       +--rw alto-server {alto-server}?
          ...
          +--rw resource* [resource-id]
             +--rw resource-id                  resource-id
             +--rw resource-type                identityref
             +--rw description?                 string
             +--rw accepted-role*               role-ref
             +--rw dependency*                  resource-ref
             +--rw alto-ird-params
             |  +--rw delegation    inet:uri
             +--rw alto-networkmap-params
             |  +--rw is-default?   boolean
             |  +--rw filtered?     boolean
             |  +---u algorithm
             +--rw alto-costmap-params
             |  +--rw filtered?             boolean
             |  +---u filter-costmap-cap
             |  +---u algorithm
             +--rw alto-endpointcost-params
             |  +---u filter-costmap-cap
             |  +---u algorithm
             +--rw alto-endpointprop-params
             |  +--rw prop-type*   endpoint-property
             |  +---u algorithm
             +--rw alto-propmap-params {propmap}?
             |  +---u algorithm
             +--rw alto-cdni-params {cdni}?
             |  +---u algorithm
             +--rw alto-update-params {incr-update}?
             |  +---u algorithm
             +--rw resource-limits
                +--rw notify-res-mem-limit?      uint64
                +--rw notify-upd-stream-limit?   uint64
                        {incr-update}?

    notifications:
      +---n alto-resource-event {alto-server}?
         +--ro resource-id                     resource-ref
         +--ro notify-res-mem-threshold?       uint64
         +--ro notify-upd-stream-threshold?    uint64 {incr-update}?
```

```
    grouping filter-costmap-cap:
      +-- cost-type-name*            cost-type-ref
      +-- cost-constraints?          boolean
      +-- max-cost-types?            uint32 {multi-cost}?
      +-- testable-cost-type-name*   cost-type-ref {multi-cost}?
      +-- calendar-attributes {cost-calendar}?
         +-- cost-type-name*        cost-type-ref
         +-- time-interval-size     decimal64
         +-- number-of-intervals    uint32
    grouping algorithm:
      +-- (algorithm)
```

Figure 8: IETF ALTO Resource Subtree Structure

A 'resource' list entry MUST include a unique 'resource-id' and a 'resource-type'.

It may also include an 'accepted-role' node containing a list of 'role-name's that is used by role-based access control for this ALTO information resource. See Section 5.4.3 for details of information resource access control.

For some 'resource-type', the 'resource' entry may also include a 'dependency' node containing the 'resource-id' of the dependent ALTO information resources (Section 9.1.5 of [RFC7285]).

For each type of ALTO information resource, the 'resource' entry may also need type-specific parameters. These type-specific parameters can be split into two categories:

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

2. The other category of the type-specific parameters is 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 an augmented 'algorithm' node. The augmented 'algorithm' node can reference data sources subscribed by the 'data-source' entries (See Section 5.4.1). An example of extending the 'algorithm' node for a specific type of 'resource' is included in Appendix A.4.

The developer does not have to 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 (Section 9.2.4 of [RFC7285]).

Each 'resource' entry may also set thresholds of memory usage and active update streams (if "incr-update" feature is enabled). Table 3 describes limits that, once exceeded, will trigger notifications to be generated:

Notification Threshold	Description
notify-res-mem-limit	Used to notify high memory utilization of the resource configured to an ALTO server instance. When exceeded, an alto-resource-event will be generated.
notify-upd-stream-limit	Used to notify a high number of active update streams that are serviced by an update resource configured to an ALTO server instance. When exceeded, an alto-resource-event will be generated.

Table 3: Notification Thresholds.

5.4.3. ALTO Information Resource Access Control Management

To satisfy R-3 in Section 4.2 and as per Section 15.5.2 of [RFC7285], the "ietf-alto" module also defines authentication and authorization related configuration to employ access control at the information resource level. The ALTO server returns the IRD to the ALTO client based on its authentication information.

The information resource access control is supported using the structure shown in Figure 9.

```
module: ietf-alto
  +--rw alto!
     ...
     +--rw alto-server {alto-server}?
        ...
        +--rw auth-client* [client-id]
        |  +--rw client-id                  string
        |  +--rw (authentication)?
        |     +--:(http)
        |     |  +--rw http-auth-client
        |     |           {http-listen,http:client-auth-supported,
        |     |            http:local-users-supported}?
        |     |     +--rw user-id    http-user-id-ref
        |     +--:(https)
        |        +--rw https-auth-client
        |        |       {http:client-auth-supported,
        |        |        http:local-users-supported}?
        |        |  +--rw user-id?   https-user-id-ref
        |        +--rw tls-auth-client
        |                {http:client-auth-supported}?
        |           +--rw ca-cert {tls:client-auth-x509-cert}?
        |           |  +---u inline-or-truststore-ca-cert-ref
        |           +--rw ee-cert {tls:client-auth-x509-cert}?
        |           |  +---u inline-or-truststore-ee-cert-ref
        |           +--rw raw-public-key
        |           |       {tls:client-auth-raw-public-key}?
        |           |  +---u inline-or-truststore-public-key-ref
        |           +--rw tls12-psks?       empty
        |           |       {tls:client-auth-tls12-psk}?
        |           +--rw tls13-epsks?      empty
        |                   {tls:client-auth-tls13-epsk}?
        +--rw role* [role-name]
        |  +--rw role-name    role-name
        |  +--rw client*      client-ref
        ...
```

Figure 9: IETF ALTO Client Authentication Subtree Structure

The structure shown in Figure 9 can be used to configure the role-based access control:

* 'auth-client' declares a list of ALTO clients that can be authenticated by the internal or external authorization server. This basic model only includes authentication approach directly provided by the HTTP or TLS server, but the operators or future documents can augment the 'authentication' choice for different authentication mechanisms.

* 'role' defines a list of roles for access control. Each role contains a list of authenticated ALTO clients. Each client can be assigned to multiple roles. The 'role-name' can be referenced by the 'accepted-role' list of a 'resource'. If an authenticated ALTO client is included in any roles with access permission to a resource, the client is granted access to that resource.

6. Design of ALTO O&M Statistics Data Model

The "ietf-alto-stats" module augments the "ietf-alto" module to include statistics at the ALTO server and information resource level (Figure 10).

```
        module: ietf-alto-stats

          augment /alto:alto/alto:alto-server:
            +--rw server-level-monitor-config
            |  +--rw time-window-size?   uint32
            +--ro server-level-stats
               +---u server-level-stats
          augment /alto:alto/alto:alto-server/alto:resource:
            +--ro resource-level-stats
               +---u resource-level-stats

          grouping server-level-stats:
            +-- discontinuity-time?    yang:timestamp
            +-- last-report-time?      yang:timestamp
            +-- num-total-req?         yang:counter64
            +-- num-total-succ?        yang:counter64
            +-- num-total-fail?        yang:counter64
            +-- num-total-last-req?    yang:gauge64
            +-- num-total-last-succ?   yang:gauge64
            +-- num-total-last-fail?   yang:gauge64
          grouping network-map-stats:
            +-- num-map-pid?   yang:gauge64
          grouping prop-map-stats:
            +-- num-map-entry?   yang:gauge64
          grouping cdni-stats:
            +-- num-base-obj?   yang:gauge64
          grouping upd-stream-stats:
            +-- num-upd-stream?           yang:gauge64
            +-- num-upd-msg-total?        yang:gauge64
            +-- num-upd-msg-max?          yang:gauge64
            +-- num-upd-msg-min?          yang:gauge64
            +-- num-upd-msg-avg?          yang:gauge64
            +-- num-upd-msg-total-last?   yang:gauge64
            +-- num-upd-msg-max-last?     yang:gauge64
            +-- num-upd-msg-min-last?     yang:gauge64
```

```
    +-- num-upd-msg-avg-last?     yang:gauge64
  grouping resource-level-stats:
    +-- discontinuity-time?    yang:timestamp
    +-- last-report-time?      yang:timestamp
    +-- num-res-upd?           yang:counter64
    +-- res-mem-size?          uint64
    +-- res-enc-size?          uint64
    +-- num-res-req?           yang:counter64
    +-- num-res-succ?          yang:counter64
    +-- num-res-fail?          yang:counter64
    +-- num-res-last-req?      yang:gauge64
    +-- num-res-last-succ?     yang:gauge64
    +-- num-res-last-fail?     yang:gauge64
    +-- network-map-stats
    |  +---u network-map-stats
    +-- endpoint-prop-stats
    |  +---u prop-map-stats
    +-- property-map-stats
    |  +---u prop-map-stats
    +-- cdni-stats
    |  +---u cdni-stats
    +-- upd-stream-stats
```

Figure 10: IETF ALTO Statistics Structure

6.1. Model for ALTO Server Failure Monitoring

To satisfy R6 in Section 4.2, the "ietf-alto-stats" module contains statistics that indicate server failures (Figure 10).

More specifically, 'num-total-*' and 'num-total-last-*' provide server-level failure counters; 'num-res-*' and 'num-res-last-*' provide information resource-level failure counters.

6.2. Model for ALTO-specific Performance Monitoring

To satisfy R7 in Section 4.2,the "ietf-alto-stats" module also contains statistics for ALTO-specific performance metrics (Figure 10).

More specifically, this data model contains the following measurement information of "system and service performance" suggested by [RFC7285] and [RFC7971]:

* Requests and responses for each information resource

* CPU and memory utilization

* ALTO map updates

* Number of PIDs

* ALTO map sizes

Besides the above measurement information suggested by [RFC7285] and [RFC7971], the "ietf-alto-stats" module also contains useful measurement information for other ALTO extensions:

* 'num-map-entry' and 'num-base-obj' provide measurement for number of generic ALTO entities (for [RFC9240] and [RFC9241])

* 'num-upd-stream' and 'num-upd-msg-*' provide statistics for update streams and messages (for [RFC8189])

The "ietf-alto-stats" module only focuses on the performance metrics that can be directly measured at the ALTO server. The following metrics for "measurement of the impact" suggested by [RFC7971] are not contained in this data model:

* Total amount and distribution of traffic

* Application performance

7. ALTO OAM YANG Modules

7.1. The "ietf-alto" YANG Module

<CODE BEGINS> file "ietf-alto@2023-02-23.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, Section 4";
  }
  import ietf-truststore {
    prefix ts;
    reference
      "RFC BBBB: A YANG Data Model for a Truststore";
  }
  import ietf-tcp-server {
    prefix tcp;
    reference
```

```
        "RFC DDDD: YANG Groupings for TCP Clients and TCP Servers";
    }
    import ietf-tls-server {
      prefix tls;
      reference
        "RFC FFFF: YANG Groupings for TLS Clients and TLS Servers";
    }
    import ietf-http-server {
      prefix http;
      reference
        "RFC GGGG: YANG Groupings for HTTP Clients and HTTP Servers";
    }

    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 a set of configured and operational
       parameters of an administrated ALTO server instance.

       Copyright (c) 2023 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 2023-02-23 {
      description
        "Initial Version.";
      reference
        "RFC XXXX: YANG Data Models for the Application-Layer Traffic
                   Optimization (ALTO) Protocol";
    }

    // Features

    feature alto-client {
      description
```

```
        "Indicates that the implementation embeds an ALTO client
         instance.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol";
    }

    feature alto-server {
      description
        "Indicates that the implementation embeds an ALTO server
         instance.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol";
    }

    feature http-listen {
      description
        "The 'http-listen' feature is only used for test deployment.
         This feature shouldn't be used in the production
         deployment.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 8.3.5 ";
    }

    feature xdom-disc {
      description
        "Indicates support of cross-domain server discovery.";
      reference
        "RFC 8686: Application-Layer Traffic Optimization (ALTO)
                   Cross-Domain Server Discovery";
    }

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

    feature incr-update {
      description
        "Indicates support of incremental update extension.";
      reference
        "RFC 8895: Application-Layer Traffic Optimization (ALTO)
                   Incremental Updates Using Server-Sent Events
```

```
                (SSE)";
    }

    feature cost-calendar {
      description
        "Indicates support of cost calendar extension.";
      reference
        "RFC 8896: Application-Layer Traffic Optimization (ALTO) Cost
                   Calendar";
    }

    feature propmap {
      description
        "Indicates support of entity property map extension.";
      reference
        "RFC 9240: An ALTO Extension: Entity Property Maps";
    }

    feature cdni {
      description
        "Indicates support of CDNi extension.";
      reference
        "RFC 9241: Content Delivery Network Interconnection (CDNI)
                   Request Routing: CDNI Footprint and Capabilities
                   Advertisement using ALTO";
    }

    feature path-vector {
      description
        "Indicates support of path vector extension.";
      reference
        "RFC 9275: An Extension for Application-Layer Traffic
                   Optimization (ALTO): Path Vector";
    }

    feature performance-metrics {
      description
        "Indicates support of performance metrics extension.";
      reference
        "RFC 9439: ALTO Performance Cost Metrics";
    }

    // Base identities

    identity resource-type {
      description
        "Base identity for type of information resource.";
      reference
```

```
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 8.1 ";
    }

    identity source-type {
      description
        "Base identity for type of data source. A data source
         indicates the origin from which the ALTO information
         resources are derived.";
    }

    identity cost-metric {
      description
        "The cost metric indicates what the cost represents.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 6.1.1";
    }

    identity cost-mode {
      description
        "The cost mode indicates how costs should be interpreted.
         Specifically, the cost mode attribute indicates whether
         indicated costs should be interpreted as numerical
         values or ordinal rankings.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 6.1.2
         RFC 9274: A Cost Mode Registry for the Application-Layer
                   Traffic Optimization (ALTO) Protocol";
    }

    identity cost-source {
      description
        "The cost source indicates the high-level type of the
         data source.";
      reference
        "RFC 9439: ALTO Performance Cost Metrics, Section 3.1";
    }

    // Identities for ALTO information resources

    identity ird {
      base resource-type;
      description
        "Identity for information resource directory.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
```

```
                Protocol, Section 9";
    }

    identity network-map {
      base resource-type;
      description
        "Identity for network map.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 5";
    }

    identity cost-map {
      base resource-type;
      description
        "Identity for cost map.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 6";
    }

    identity endpoint-prop {
      base resource-type;
      description
        "Identity for endpoint property service.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 11.4.1";
    }

    identity endpoint-cost {
      base resource-type;
      description
        "Identity for endpoint cost service.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 11.5.1";
    }

    identity property-map {
      base resource-type;
      description
        "Identity for property map.";
      reference
        "RFC 9240: An ALTO Extension: Entity Property Maps";
    }

    identity cdni {
```

```
    base resource-type;
    description
      "Identity for Content Delivery Network Interconnection (CDNI)
       advertisement service.";
    reference
      "RFC 9241: Content Delivery Network Interconnection (CDNI)
                 Request Routing: CDNI Footprint and Capabilities
                 Advertisement using ALTO";
  }

  identity update {
    base resource-type;
    description
      "Identity for update stream service.";
    reference
      "RFC 8895: Application-Layer Traffic Optimization (ALTO)
                 Incremental Updates Using Server-Sent Events
                 (SSE)";
  }

  // Identities for cost mode

  identity numerical {
    base cost-mode;
    description
      "This mode indicates that it is safe to perform numerical
       operations";
    reference
      "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                 Protocol, Section 6.1.2.1";
  }

  identity ordinal {
    base cost-mode;
    description
      "This mode indicates that the cost values in a cost map
       represent ranking";
    reference
      "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                 Protocol, Section 6.1.2.2";
  }

  identity array {
    if-feature "path-vector";
    base cost-mode;
    description
      "This mode indicates that every cost value in the response
       body of a (Filtered) Cost Map or an Endpoint Cost Service is
```

```
          interpreted as a JSON array.";
      reference
        "RFC 9275: An Extension for Application-Layer Traffic
                   Optimization (ALTO): Path Vector, Section 6.5.2";
    }

    // Identities for cost metrics

    identity routingcost {
      base cost-metric;
      description
        "This metric conveys a generic measure for the cost of
         routing traffic from a source to a destination.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 6.1.1.1";
    }

    identity ane-path {
      if-feature "path-vector";
      base cost-metric;
      description
        "This metric indicates that the value of such a cost type
         conveys an array of Abstract Network Element (ANE) names,
         where each ANE name uniquely represents an ANE traversed by
         traffic from a source to a destination.";
      reference
        "RFC 9275: An Extension for Application-Layer Traffic
                   Optimization (ALTO): Path Vector, Section 6.5.1";
    }

    identity delay-ow {
      if-feature "performance-metrics";
      base cost-metric;
      description
        "One-way delay.";
      reference
        "RFC 9439: ALTO Performance Cost Metrics, Section 4.1";
    }

    identity delay-rt {
      if-feature "performance-metrics";
      base cost-metric;
      description
        "Round-trip delay.";
      reference
        "RFC 9439: ALTO Performance Cost Metrics, Section 4.2";
    }
```

```
    identity delay-variation {
      if-feature "performance-metrics";
      base cost-metric;
      description
        "Delay variation.";
      reference
        "RFC 9439: ALTO Performance Cost Metrics, Section 4.3";
    }

    identity lossrate {
      if-feature "performance-metrics";
      base cost-metric;
      description
        "Loss rate.";
      reference
        "RFC 9439: ALTO Performance Cost Metrics, Section 4.4";
    }

    identity hopcount {
      if-feature "performance-metrics";
      base cost-metric;
      description
        "Hop count.";
      reference
        "RFC 9439: ALTO Performance Cost Metrics, Section 4.5";
    }

    identity tput {
      if-feature "performance-metrics";
      base cost-metric;
      description
        "TCP throughput.";
      reference
        "RFC 9439: ALTO Performance Cost Metrics, Section 5.1";
    }

    identity bw-residual {
      if-feature "performance-metrics";
      base cost-metric;
      description
        "Residual bandwidth.";
      reference
        "RFC 9439: ALTO Performance Cost Metrics, Section 5.2";
    }

    identity bw-available {
      if-feature "performance-metrics";
      base cost-metric;
```

```
      description
        "Available bandwidth.";
      reference
        "RFC 9439: ALTO Performance Cost Metrics, Section 5.3";
    }

    // Identities for cost sources

    identity nominal {
      if-feature "performance-metrics";
      base cost-source;
      description
        "The 'nominal' category indicates that the metric value is
         statically configured by the underlying devices.";
      reference
        "RFC 9439: ALTO Performance Cost Metrics, Section 3.1";
    }

    identity sla {
      if-feature "performance-metrics";
      base cost-source;
      description
        "The 'sla' category indicates that the metric value is
         derived from some commitment which this document refers to
         as service-level agreement (SLA).";
      reference
        "RFC 9439: ALTO Performance Cost Metrics, Section 3.1";
    }

    identity estimation {
      if-feature "performance-metrics";
      base cost-source;
      description
        "The 'estimation' category indicates that the metric value is
         computed through an estimation process.";
      reference
        "RFC 9439: ALTO Performance Cost Metrics, Section 3.1";
    }

    // Typedefs

    typedef resource-id {
      type string {
        length "1..64";
        pattern '[0-9a-zA-Z\-:@_]*';
      }
      description
        "Type for a resource ID that are used to reference an ALTO
```

```
          information resource.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 9.1.1";
    }

    typedef cost-type-name {
      type string {
        length "1..max";
      }
      description
        "Type for the name of a single CostType that can be
         referenced by other ALTO information resources.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 9.2.2";
    }

    typedef meta-key {
      type string {
        length "1..max";
      }
      description
        "Type for a custom meta key of an ALTO server.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 8.4.1";
    }

    typedef endpoint-property {
      type union {
        type resource-specific-endpoint-property;
        type global-endpoint-property;
      }
      description
        "Type for an endpoint property.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 10.8";
    }

    typedef resource-specific-endpoint-property {
      type string {
        length "1..97";
        pattern '[0-9a-zA-Z\-:@_]*\.[0-9a-zA-Z\-:_]*';
      }
      description
        "Type for a resource-specific endpoint property.";
```

```
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 10.8.1";
    }

    typedef global-endpoint-property {
      type string {
        length "1..32";
        pattern '[0-9a-zA-Z\-:_]*';
      }
      description
        "Type for a global endpoint property.";
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 10.8.2";
    }

    typedef source-id {
      type string {
        length "1..max";
      }
      description
        "Type for a data source ID that are used to reference a data
         source.";
    }

    typedef role-name {
      type string {
        length "1..max";
      }
      description
        "Type for a name of a role for role-based access control.";
    }

    // Typedefs for referencing purposes

    typedef cost-type-ref {
      type leafref {
        path "/alto:alto/alto:alto-server/alto:cost-type"
           + "/alto:cost-type-name";
      }
      description
        "Type to reference a cost type name.";
    }

    typedef data-source-ref {
      type leafref {
        path "/alto:alto/alto:alto-server/alto:data-source"
```

```
          + "/alto:source-id";
    }
    description
      "Type to reference a data source identifier.";
  }

  typedef http-user-id-ref {
    type leafref {
      path "/alto:alto/alto:alto-server/alto:listen"
         + "/alto:http/alto:http-server-parameters"
         + "/alto:client-authentication/alto:users"
         + "/alto:user/alto:user-id";
    }
    description
      "Type to reference an HTTP client user id.";
  }

  typedef https-user-id-ref {
    type leafref {
      path "/alto:alto/alto:alto-server/alto:listen"
         + "/alto:https/alto:http-server-parameters"
         + "/alto:client-authentication/alto:users"
         + "/alto:user/alto:user-id";
    }
    description
      "Type to reference an HTTPS client user id.";
  }

  typedef inline-ca-cert-ref {
    type leafref {
      path "/alto:alto/alto:alto-server/alto:listen"
         + "/alto:https/alto:tls-server-parameters"
         + "/alto:client-authentication/alto:ca-certs"
         + "/alto:inline-definition/alto:certificate"
         + "/alto:name";
    }
    description
      "Type to reference a TLS CA certificate.";
  }

  typedef inline-ee-cert-ref {
    type leafref {
      path "/alto:alto/alto:alto-server/alto:listen"
         + "/alto:https/alto:tls-server-parameters"
         + "/alto:client-authentication/alto:ee-certs"
         + "/alto:inline-definition/alto:certificate"
         + "/alto:name";
    }
```

```
      description
        "Type to reference a TLS EE certificate.";
    }

    typedef inline-raw-public-key-ref {
      type leafref {
        path "/alto:alto/alto:alto-server/alto:listen"
           + "/alto:https/alto:tls-server-parameters"
           + "/alto:client-authentication/alto:raw-public-keys"
           + "/alto:inline-definition/alto:public-key"
           + "/alto:name";
      }
      description
        "Type to reference a raw pubic key.";
    }

    typedef resource-ref {
      type leafref {
        path "/alto:alto/alto:alto-server/alto:resource"
           + "/alto:resource-id";
      }
      description
        "Type to reference a resource identifier.";
    }

    typedef role-ref {
      type leafref {
        path "/alto:alto/alto:alto-server/alto:role"
           + "/alto:role-name";
      }
      description
        "Type to reference a role.";
    }

    typedef client-ref {
      type leafref {
        path "/alto:alto/alto:alto-server/alto:auth-client"
           + "/alto:client-id";
      }
      description
        "Type to reference an authenticated client.";
    }

    // Groupings

    grouping filter-costmap-cap {
      description
        "This grouping defines a data model for
```

```
          FilteredCostMapCapabilities.";
        reference
          "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                     Protocol, Section 11.3.2.4";
        leaf-list cost-type-name {
          type cost-type-ref;
          min-elements 1;
          description
            "Supported cost types.";
        }
        leaf cost-constraints {
          type boolean;
          default false;
          description
            "If set to true, then the ALTO server allows cost
             constraints to be included in requests to the
             corresponding URI.";
        }
        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 and can return a
             multi-cost map with any combination of N or
             fewer cost types in the 'cost-type-names' list.";
        }
        leaf-list testable-cost-type-name {
          if-feature "multi-cost";
          type cost-type-ref;
          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";
          description
            "Configuration for CalendarAttributes.";
          reference
            "RFC 8896: Application-Layer Traffic Optimization (ALTO)
                       Cost Calendar, Section 4.1";
          leaf-list cost-type-name {
            type cost-type-ref;
            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;
          }
          units "seconds";
          mandatory true;
          description
            "The duration of an ALTO Calendar time interval.";
        }
        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.";
        }
      }
    }

    grouping algorithm {
      description
        "This grouping defines the base data model for information
         resource creation algorithm.";
      choice algorithm {
        mandatory true;
        description
          "Information resource creation algorithm to be augmented.";
      }
      reference
        "RFC XXXX: YANG Data Models for the Application-Layer Traffic
                   Optimization (ALTO) Protocol, Section 5.4.2";
    }

    grouping alto-server {
      description
        "A reuseable grouping for configuring an ALTO server without
         any consideration for how underlying transport sessions are
         established.";
      leaf base-uri {
        type inet:uri;
        description
          "The base URI for the ALTO server.";
      }
```

```
    }

    grouping alto-server-listen-stack {
      description
        "A reuseable grouping for configuring an ALTO server
         'listen' protocol stack for a single connection.";
      choice transport {
        mandatory true;
        description
          "Selects between available transports.";
        case http {
          if-feature "http-listen";
          container http {
            description
              "Configures ALTO server stack assuming that
               TLS-termination is handled externally.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP server parameters
                 to avoid name collisions.";
              uses tcp:tcp-server-grouping {
                refine "local-port" {
                  default "80";
                  description
                    "The RESTCONF server will listen on the IANA-
                     assigned well-known port value for 'http' (80)
                     if no value is specified.";
                }
              }
            }
            container http-server-parameters {
              description
                "A wrapper around the HTTP server parameters
                 to avoid name collisions.";
              uses http:http-server-grouping;
            }
            container alto-server-parameters {
              description
                "A wrapper around the ALTO server parameters
                 to avoid name collisiions.";
              uses alto-server;
            }
          }
        }
        case https {
          container https {
            description
              "Configures ALTO server stack assuming that
```

```
               TLS-termination is handled internally.";
            container tcp-server-parameters {
              description
                "A wrapper around the TCP server parameters
                 to avoid name collisions.";
              uses tcp:tcp-server-grouping {
                refine "local-port" {
                  default "443";
                  description
                    "The ALTO server will listen on the IANA-
                     assigned well-known port value for 'https'
                     (443) if no value is specified.";
                }
              }
            }
            container tls-server-parameters {
              description
                "A wrapper around the TLS server parameters
                 to avoid name collisions.";
              uses tls:tls-server-grouping;
            }
            container http-server-parameters {
              description
                "A wrapper around the HTTP server parameters
                 to avoid name collisions.";
              uses http:http-server-grouping;
            }
            container alto-server-parameters {
              description
                "A wrapper around the ALTO server parameters
                 to avoid name collisions.";
              uses alto-server;
            }
          }
        }
      }
    }

    grouping alto-server-discovery {
      description
        "Grouping for the configuration of how to set up server
         discovery for clients or other ALTO servers to discovery the
         URI of this ALTO server.";
      choice method {
        description
          "Selects among available server discovery methods.";
        case reverse-dns {
          if-feature "xdom-disc";
```

```
          description
            "Configures DNS NAPTR records for cross-domain ALTO server
             discovery using reverse DNS lookup.";
          reference
            "RFC 8686: Application-Layer Traffic Optimization (ALTO)
                       Cross-Domain Server Discovery.";
          container rdns-naptr-records {
            description
              "Configuration parameters for DNS NAPTR records.";
            leaf-list static-prefix {
              type inet:ip-prefix;
              description
                "Specifies a list of static IP prefixes.";
            }
            leaf-list dynamic-prefix-source {
              type data-source-ref;
              description
                "Dynamic IP prefixes collected from data sources.";
            }
          }
        }
      }
    }

    grouping alto-server-discovery-client {
      description
        "Grouping for configuration of how a client can discover
         an ALTO server.";
      choice method {
        description
          "Selects among available server discovery methods.";
        case reverse-dns {
          if-feature "xdom-disc";
          description
            "Uses reverse DNS lookup to discover an ALTO server.";
          reference
            "RFC 8686: Application-Layer Traffic Optimization (ALTO)
                       Cross-Domain Server Discovery.";
          container rdns-params {
            description
              "Defines a set of parameters for reverse DNS
               lookups.";
            leaf-list dns-server {
              type inet:host;
              description
                "Provides a DNS server list for reverse DNS lookup.";
            }
          }
```

```
        }
      }
    }

    grouping alto-logging-system {
      description
        "Grouping for configuration of logging system used by the
         ALTO server.";
      choice logging-system {
        description
          "Selects among available logging systems.";
        case syslog {
          description
            "Specifies syslog as the logging system.";
          container syslog-params {
            description
              "Provides a set of syslog parameters.";
            leaf config-file {
              type inet:uri {
                pattern 'file:.*';
              }
              description
                "Indicates the file location of the syslog
                 configuration.";
            }
          }
        }
      }
    }

    grouping inline-or-truststore-ca-cert-ref {
      description
        "Grouping for the reference of a CA certificate to
         authenticate the TLS client.";
      choice inline-or-truststore {
        description
          "Selects between inline and truststore";
        case inline {
          if-feature "ts:inline-definitions-supported";
          leaf inline {
            type inline-ca-cert-ref;
            description
              "Reference to an inline CA certificate configured by
               the TLS server.";
          }
          description
            "Reference of an inline CA certificate to authenticate
             the TLS client.";
```

```
        }
        case central-truststore {
          if-feature "ts:central-truststore-supported";
          if-feature "ts:certificates";
          uses ts:certificate-ref-grouping;
          description
            "Reference of a CA certificate in the truststore to
             authenticate the TLS client.";
        }
      }
    }

    grouping inline-or-truststore-ee-cert-ref {
      description
        "Grouping for the reference of a EE certificate to
         authenticate the TLS client.";
      choice inline-or-truststore {
        description
          "Selects between inline and truststore";
        case inline {
          if-feature "ts:inline-definitions-supported";
          leaf inline {
            type inline-ee-cert-ref;
            description
              "Reference to an inline EE certificate configured by
               the TLS server.";
          }
          description
            "Reference of an inline EE certificate to authenticate
             the TLS client.";
        }
        case central-truststore {
          if-feature "ts:central-truststore-supported";
          if-feature "ts:certificates";
          uses ts:certificate-ref-grouping;
          description
            "Reference of a EE certificate in the truststore to
             authenticate the TLS client.";
        }
      }
    }

    grouping inline-or-truststore-public-key-ref {
      description
        "Grouping for the reference of a raw public key to
         authenticate the TLS client.";
      choice inline-or-truststore {
        description
```

```
          "Selects between inline and truststore";
        case inline {
          if-feature "ts:inline-definitions-supported";
          leaf inline {
            type inline-raw-public-key-ref;
            description
              "Reference to an inline public key configured by the
               TLS server.";
          }
          description
            "Reference of an inline public key to authenticate the
             TLS client.";
        }
        case central-truststore {
          if-feature "ts:central-truststore-supported";
          if-feature "ts:public-keys";
          uses ts:public-key-ref-grouping;
          description
            "Reference of a raw public key in the truststore to
             authenticate the TLS client.";
        }
      }
    }

    // Top-level container

    container alto {
      presence "The ALTO service is enabled";
      description
        "Indicates a set of parameters for both ALTO clients and
         servers. A single device can implement either alto-client or
         alto-server. No need to implement both.";
      list alto-client {
        if-feature alto-client;
        key "client-id";
        description
          "The ALTO client configuration.";
        leaf client-id {
          type string;
          description
            "A unique identifier of a client that can be referenced
             by a data source or a resource creation algorithm to
             communicate with other ALTO servers.";
        }
        container server-discovery {
          description
            "Specifies a set of parameters for ALTO server discovery.";
          uses alto-server-discovery-client;
```

```
        }
      }
      container alto-server {
        if-feature alto-server;
        description
          "The ALTO server instance configuration.";
        container listen {
          description
            "Configure the ALTO server to listen for ALTO clients.";
          uses alto-server-listen-stack;
        }
        container server-discovery {
          description
            "Configures how the ALTO server to be discovered by
             others.";
          uses alto-server-discovery;
        }
        container logging-system {
          description
            "Configure logging system to capture log messages
             generated by the ALTO server.";
          uses alto-logging-system;
        }
        list cost-type {
          key "cost-type-name";
          description
            "Mapping between name and referenced cost type.";
          leaf cost-type-name {
            type cost-type-name;
            description
              "The name to reference a cost type.";
          }
          leaf cost-mode {
            type identityref {
              base cost-mode;
            }
            mandatory true;
            description
              "The referenced cost mode.";
          }
          leaf cost-metric {
            type identityref {
              base cost-metric;
            }
            mandatory true;
            description
              "The referenced cost metric.";
          }
```

```
          leaf description {
            type string;
            description
              "A human-readable description fo the 'cost-mode' and
               'cost-metric'.";
          }
          container cost-context {
            if-feature "performance-metrics";
            description
              "Context of how the metric is obtained.";
            leaf cost-source {
              type identityref {
                base cost-source;
              }
              mandatory true;
              description
                "The referenced cost source.";
            }
            container parameters {
              description
                "Additional computation parameters for the cost
                 source.";
              choice parameters {
                description
                  "Cases of parameters to be augmented.";
              }
            }
          }
        }
        list meta {
          key "meta-key";
          description
            "Mapping of custom meta information";
          reference
            "RFC 7285: Application-Layer Traffic Optimization
                       (ALTO) Protocol, Section 8.4.1";
          leaf meta-key {
            type meta-key;
            description
              "Custom meta key";
          }
          leaf meta-value {
            type binary;
            mandatory true;
            description
              "Custom meta value encoded with the base64 encoding
               schema. The encoded value must be a valid JSON
               value.";
```

```
          }
        }
        list auth-client {
          key "client-id";
          description
            "List of authenticated ALTO clients.";
          leaf client-id {
            type string;
            description
              "Identifier to reference an ALTO client.";
          }
          choice authentication {
            description
              "Choice of authentication methods to identify this
               ALTO client. If no authentication method is
               configured, the client must be ignored.";
            case http {
              description
                "The client is authenticated by the HTTP server.";
              container http-auth-client {
                if-feature "http-listen";
                if-feature "http:client-auth-supported";
                if-feature "http:local-users-supported";
                description
                  "Parameters of the authenticated HTTP client.";
                leaf user-id {
                  type http-user-id-ref;
                  mandatory true;
                  description
                    "Reference of the user-id for the authenticated
                     HTTP client.";
                }
              }
            }
            case https {
              description
                "The client is authenticated by the HTTP server.";
              container https-auth-client {
                if-feature "http:client-auth-supported";
                if-feature "http:local-users-supported";
                description
                  "Parameters to identify an authenticated HTTPS
                   client.";
                leaf user-id {
                  type https-user-id-ref;
                  description
                    "Reference of the user-id for the authenticated
                     HTTPS client.";
```

```
            }
          }
          container tls-auth-client {
            if-feature "tls:client-auth-supported";
            description
              "Parameters to identify na authenticated TLS
               client.";
            container ca-cert {
              if-feature "tls:client-auth-x509-cert";
              description
                "Reference of the CA certificate to authenticate
                 the TLS client.";
              uses inline-or-truststore-ca-cert-ref;
            }
            container ee-cert {
              if-feature "tls:client-auth-x509-cert";
              description
                "Reference of the EE certificate to authenticate
                 the TLS client.";
              uses inline-or-truststore-ee-cert-ref;
            }
            container raw-public-key {
              if-feature "tls:client-auth-raw-public-key";
              description
                "Reference of the raw public key to authenticate
                 the TLS client.";
              uses inline-or-truststore-public-key-ref;
            }
            leaf tls12-psks {
              if-feature "tls:client-auth-tls12-psk";
              type empty;
              description
                "Identicate that the client is authenticated by
                 the TLS server using the configured PSKs
                 (pre-shared or pairwise-symmetric keys).";
            }
            leaf tls13-epsks {
              if-feature "tls:client-auth-tls13-epsk";
              type empty;
              description
                "Identicate that the client is authenticated by
                 the TLS 1.3 server using the configured external
                 PSKs (pre-shared keys).";
            }
          }
        }
      }
    }
```

```
        list role {
          key "role-name";
          description
            "List of roles for access control.";
          leaf role-name {
            type role-name;
            description
              "Name of a role for access control.";
          }
          leaf-list client {
            type client-ref;
            description
              "List of authenticated ALTO clients assigned to the
               role.";
          }
        }
        list data-source {
          key "source-id";
          description
            "List of subscribed data sources.";
          leaf source-id {
            type source-id;
            description
              "Data source id that can be referenced by information
               resource creation algorithms.";
          }
          leaf source-type {
            type identityref {
              base source-type;
            }
            mandatory true;
            description
              "Identify the type of the data source.";
          }
          container source-params {
            description
              "Data source specific configuration.";
            reference
              "RFC XXXX: YANG Data Models for the Application-Layer
                         Traffic Optimization (ALTO) Protocol,
                         Section 5.4.1";
          }
        }
        list resource {
          key "resource-id";
          description
            "ALTO information resources to be defined";
          leaf resource-id {
```

```
            type resource-id;
            description
              "resource-id to be defined.";
          }
          leaf resource-type {
            type identityref {
              base resource-type;
            }
            mandatory true;
            description
              "identityref to be defined.";
          }
          leaf description {
            type string;
            description
              "The optional description for this information
               resource.";
          }
          leaf-list accepted-role {
            type role-ref;
            description
              "Roles allowed to access this information resource.";
          }
          leaf-list dependency {
            type resource-ref;
            description
              "A list of dependent information resources.";
          }
          container alto-ird-params {
            when 'derived-from-or-self(../resource-type,'
               + '"alto:ird")';
            description
              "IRD-specific configuration.";
            leaf delegation {
              type inet:uri;
              mandatory true;
              description
                "Upstream IRD to be delegated.";
            }
          }
          container alto-networkmap-params {
            when 'derived-from-or-self(../resource-type,'
               + '"alto:network-map")';
            description
              "Filtered Network Map specific configuration.";
            reference
              "RFC 7285: Application-Layer Traffic Optimization
                         (ALTO) Protocol, Sections 11.2.1 and
```

```
                    11.3.1";
          leaf is-default {
            type boolean;
            description
              "Sets whether this is the default network map.";
          }
          leaf filtered {
            type boolean;
            default false;
            description
              "Configures whether filtered network map is
               supported.";
          }
          uses algorithm;
        }
        container alto-costmap-params {
          when 'derived-from-or-self(../resource-type,'
             + '"alto:cost-map")';
          description
            "Filtered Cost Map specific configuration.";
          reference
            "RFC 7285: Application-Layer Traffic Optimization
                       (ALTO) Protocol, Sections 11.2.2 and
                       11.3.2";
          leaf filtered {
            type boolean;
            description
              "Configures whether filtered cost map is supported.";
          }
          uses filter-costmap-cap;
          uses algorithm;
        }
        container alto-endpointcost-params {
          when 'derived-from-or-self(../resource-type,'
             + '"alto:endpoint-cost")';
          description
            "Endpoint Cost Service specific configuration.";
          reference
            "RFC 7285: Application-Layer Traffic Optimization
                       (ALTO) Protocol, Section 11.5";
          uses filter-costmap-cap;
          uses algorithm;
        }
        container alto-endpointprop-params {
          when 'derived-from-or-self(../resource-type,'
             + '"alto:endpoint-prop")';
          description
            "Endpoint Cost Service specific configuration.";
```

```
        reference
          "RFC 7285: Application-Layer Traffic Optimization
                     (ALTO) Protocol, Section 11.5";
        leaf-list prop-type {
          type endpoint-property;
          min-elements 1;
          description
            "Supported endpoint properties.";
        }
        uses algorithm;
      }
      container alto-propmap-params {
        when 'derived-from-or-self(../resource-type,'
           + '"alto:property-map")';
        if-feature "propmap";
        description
          "(Filtered) Entity Property Map specific
           configuration.";
        reference
          "RFC 9240: An ALTO Extension: Entity Property Maps";
        uses algorithm;
      }
      container alto-cdni-params {
        when 'derived-from-or-self(../resource-type,'
           + '"alto:cdni")';
        if-feature "cdni";
        description
          "CDNi specific configuration.";
        reference
          "RFC 9241: Content Delivery Network Interconnection
                     (CDNI) Request Routing: CDNI Footprint and
                     Capabilities Advertisement using ALTO";
        uses algorithm;
      }
      container alto-update-params {
        when 'derived-from-or-self(../resource-type,'
           + '"alto:update")';
        if-feature "incr-update";
        description
          "Incremental Updates specific configuration.";
        reference
          "RFC 8895: Application-Layer Traffic Optimization
                     (ALTO) Incremental Updates Using Server-Sent
                     Events (SSE)";
        uses algorithm;
      }
      container resource-limits {
        description
```

```
              "Sets resource limits.";
            leaf notify-res-mem-limit {
              type uint64;
              units "bytes";
              description
                "Notification of resource memory usage.

                 Notification must be generated when the defined
                 threshold is reached.";
            }
            leaf notify-upd-stream-limit {
              when 'derived-from-or-self(../../resource-type,'
                 + '"alto:update")';
              if-feature "incr-update";
              type uint64;
              description
                "Notification of number of active update streams.

                 Notification must be generated when the defined
                 threshold is reached.";
            }
          }
        }
      }
    }

    // Notifications

    notification alto-resource-event {
      if-feature alto-server;
      description
        "Notifications must be generated when notify-res-mem-limit
         and/or notify-upd-stream-limit thresholds are reached.";
      leaf resource-id {
        type resource-ref;
        mandatory true;
        description
          "Resource identifier.";
      }
      leaf notify-res-mem-threshold {
        type uint64;
        units "bytes";
        description
          "The notify-res-mem-limit threshold has been fired.";
      }
      leaf notify-upd-stream-threshold {
        if-feature "incr-update";
        type uint64;
```

```
        description
          "The notify-upd-stream-limit threshold has been fired.";
      }
    }
  }
  <CODE ENDS>
```

7.2. The "ietf-alto-stats" YANG Module

```
  <CODE BEGINS> file "ietf-alto-stats@2023-02-23.yang"
  module ietf-alto-stats {
    yang-version 1.1;
    namespace "urn:ietf:params:xml:ns:yang:ietf-alto-stats";
    prefix alto-stats;

    import ietf-yang-types {
      prefix yang;
      reference
        "RFC 6991: Common YANG Data Types, Section 3";
    }
    import ietf-alto {
      prefix alto;
      reference
        "RFC XXXX: YANG Data Models for the Application-Layer
                   Traffic Optimization (ALTO) Protocol";
    }

    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 a set of statistics of an ALTO
       server instance.

       Copyright (c) 2023 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 2023-02-23 {
      description
        "Initial Version.";
      reference
        "RFC XXXX: YANG Data Models for the Application-Layer
                   Traffic Optimization (ALTO) Protocol";
    }

    // Groupings

    grouping server-level-stats {
      description
        "This grouping defines statistics for server-level
         monitoring.";
      leaf discontinuity-time {
        type yang:timestamp;
        description
          "The time on the most recent occasion at which the ALTO
           server suffered a discontinuity. This must be initialized
           when the ALTO server is configured or rebooted.";
      }
      leaf last-report-time {
        type yang:timestamp;
        description
          "The time on the most recent occasion at which the
           statistics were reported.";
      }
      leaf num-total-req {
        type yang:counter64;
        description
          "The total number of ALTO requests received by the ALTO
           server.";
      }
      leaf num-total-succ {
        type yang:counter64;
        description
          "The total number of successful responses sent by the ALTO
           server.";
      }
      leaf num-total-fail {
        type yang:counter64;
        description
          "The total number of failed responses sent by the ALTO
           server.";
      }
      leaf num-total-last-req {
```

```
        type yang:gauge64;
        description
          "The total number of ALTO requests received by the ALTO
           server within the last time window. The duration of the
           time window is configured by time-window-size parameter.";
      }
      leaf num-total-last-succ {
        type yang:gauge64;
        description
          "The total number of successful responses sent by the ALTO
           server within the last time window. The duration of the
           time window is configured by time-window-size parameter.";
      }
      leaf num-total-last-fail {
        type yang:gauge64;
        description
          "The total number of failed responses sent by the ALTO
           server within the last time window. The duration of the
           time window is configured by time-window-size parameter.";
      }
    }

    grouping network-map-stats {
      description
        "This grouping defines resource-specific statstics for the
         network map service only.";
      leaf num-map-pid {
        type yang:gauge64;
        description
          "Number of PIDs contained in the network map.";
      }
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 5";
    }

    grouping prop-map-stats {
      description
        "This grouping defines resource-specific statstics for the
         endpoint property or property map service only.";
      leaf num-map-entry {
        type yang:gauge64;
        description
          "Number of ALTO entities contained in the property map.";
      }
      reference
        "RFC 7285: Application-Layer Traffic Optimization (ALTO)
                   Protocol, Section 11.4.1
```

```
          RFC 9240: An ALTO Extension: Entity Property Maps";
      }

      grouping cdni-stats {
        description
          "This grouping defines resource-specific statstics for the
           CDNI advertisement service only.";
        leaf num-base-obj {
          type yang:gauge64;
          description
            "Number of base CDNi advertisement objects contained in the
             CDNI resource.";
        }
        reference
          "RFC 9241: Content Delivery Network Interconnection (CDNI)
                     Request Routing: CDNI Footprint and Capabilities
                     Advertisement using ALTO";
      }

      grouping upd-stream-stats {
        description
          "This grouping defines resource-specific statstics for the
           update stream service only.";
        leaf num-upd-stream {
          type yang:gauge64;
          description
            "Number of active update streams connected to the update
             stream service.";
        }
        leaf num-upd-msg-total {
          type yang:gauge64;
          description
            "Total number of update messages sent to all the active
             update streams.";
        }
        leaf num-upd-msg-max {
          type yang:gauge64;
          description
            "The maximum value over the total number of update messages
             sent to each active update stream. Assume there are 3
             active update streams A, B, and C with 4, 3, and 2 update
             messages sent to them respectively, the value of this
             metric is 4. After a while, if there is no new update
             message sent to any update stream, but the update stream A
             is closed, then the value of this metric is updated to
             3.";
        }
        leaf num-upd-msg-min {
```

```
        type yang:gauge64;
        description
          "The minimum value over the total number of update messages
           sent to each active update stream. The procedure is similar
           to num-msg-max.";
      }
      leaf num-upd-msg-avg {
        type yang:gauge64;
        description
          "The average value over the total number of update messages
           sent to each active update stream. The procedure is similar
           to num-msg-max.";
      }
      leaf num-upd-msg-total-last {
        type yang:gauge64;
        description
          "Total number of update messages sent to all the active
           update streams within the last time window. The duration
           of the time window is configured by time-window-size
           parameter.";
      }
      leaf num-upd-msg-max-last {
        type yang:gauge64;
        description
          "The maximum value over the number of update messages sent
           to each active update stream within the last time window.
           The procedure is similar to num-msg-max, but only count
           the update messages within the last time window. The
           duration of the time window is configured by
           time-window-size parameter.";
      }
      leaf num-upd-msg-min-last {
        type yang:gauge64;
        description
          "The minimal value over the number of update messages sent
           to each active update stream within the last time window.
           The procedure is similar to num-msg-max, but only count
           the update messages within the last time window. The
           duration of the time window is configured by
           time-window-size parameter.";
      }
      leaf num-upd-msg-avg-last {
        type yang:gauge64;
        description
          "The average value over the number of update messages sent
           to each active update stream within the last time window.
           The procedure is similar to num-msg-max, but only count
           the update messages within the last time window. The
```

```
            duration of the time window is configured by
            time-window-size parameter.";
        }
        reference
          "RFC 8895: Application-Layer Traffic Optimization (ALTO)
                     Incremental Updates Using Server-Sent Events
                     (SSE)";
      }

      grouping resource-level-stats {
        description
          "This grouping defines statistics for resource-level
           monitoring.";
        leaf discontinuity-time {
          type yang:timestamp;
          description
            "The time on the most recent occasion at which the ALTO
             service providing the information resource suffered a
             discontinuity. This must be initialized when the ALTO
             information resource is configured or the ALTO server is
             rebooted.";
        }
        leaf last-report-time {
          type yang:timestamp;
          description
            "The time on the most recent occasion at which the
             statistics are reported.";
        }
        leaf num-res-upd {
          type yang:counter64;
          description
            "The number of version updates since the information
             resource was created.";
        }
        leaf res-mem-size {
          type uint64;
          units "bytes";
          description
            "Memory size utilized by the information resource.";
        }
        leaf res-enc-size {
          type uint64;
          units "bytes";
          description
            "Size of JSON encoded data of the information resource.";
        }
        leaf num-res-req {
          type yang:counter64;
```

```
        description
          "The total number of ALTO requests to this information
           resource.";
      }
      leaf num-res-succ {
        type yang:counter64;
        description
          "The total number of successful responses for requests to
           this information resource.";
      }
      leaf num-res-fail {
        type yang:counter64;
        description
          "The total number of failed responses for requests to this
           information resource.";
      }
      leaf num-res-last-req {
        type yang:gauge64;
        description
          "The number of ALTO requests to this information resource
           within the last time window. The duration of the time
           window is configured by time-window-size parameter.";
      }
      leaf num-res-last-succ {
        type yang:gauge64;
        description
          "The number of successful responses for requests to this
           information resource within the last time window. The
           duration of the time window is configured by
           time-window-size parameter.";
      }
      leaf num-res-last-fail {
        type yang:gauge64;
        description
          "The number of failed responses for requests to this
           information resource within the last time window. The
           duration of the time window is configured by
           time-window-size parameter.";
      }
      container network-map-stats {
        when 'derived-from-or-self(../../alto:resource-type,'
           + '"alto:network-map")';
        description
          "Resource-specific statistics for network map
           service only.";
        uses network-map-stats;
      }
      container endpoint-prop-stats {
```

```
        when 'derived-from-or-self(../../alto:resource-type,'
           + '"alto:endpoint-prop")';
        description
          "Resource-specific statistics for endpoint property
           service only.";
        uses prop-map-stats;
      }
      container property-map-stats {
        when 'derived-from-or-self(../../alto:resource-type,'
           + '"alto:property-map")';
        description
          "Resource-specific statistics for entity property map
           service only.";
        uses prop-map-stats;
      }
      container cdni-stats {
        when 'derived-from-or-self(../../alto:resource-type,'
           + '"alto:cdni")';
        description
          "Resource-specific statistics for CDNI advertisement
           service only.";
        uses cdni-stats;
      }
      container upd-stream-stats {
        when 'derived-from-or-self(../../alto:resource-type,'
           + '"alto:update")';
        description
          "Resource-specific statistics for update stream service
           only.";
        uses upd-stream-stats;
      }
    }

    // Augment modules to add statistics

    augment "/alto:alto/alto:alto-server" {
      description
        "Augmenting statistics and configuration parameters for
         server-level monitoring.";
      container server-level-monitor-config {
        description
          "Configuration parameters for server-level monitoring.";
        leaf time-window-size {
          type uint32;
          units "seconds";
          default "300";
          description
            "Duration of the time window within that the statistics
```

```
              are reported.";
          }
        }
        container server-level-stats {
          config false;
          description
            "Top-level statistics for the whole ALTO server.";
          uses server-level-stats;
        }
      }

      augment "/alto:alto/alto:alto-server/alto:resource" {
        description
          "Augmenting statistics and configuration parameters for
           resource-level monitoring.";
        container resource-level-stats {
          config false;
          description
            "Common statistics for each information resource.";
          uses resource-level-stats;
        }
      }
    }
    <CODE ENDS>
```

8. Security Considerations

The "ietf-alto" and "ietf-alto-stats" YANG modules specified in this document define a schema for data that is designed to be accessed via network management protocols such as NETCONF [RFC6241] or RESTCONF [RFC8040]. The lowest NETCONF layer is the secure transport layer, and the mandatory-to-implement secure transport is Secure Shell (SSH) [RFC6242]. The lowest RESTCONF layer is HTTPS, and the mandatory-to-implement secure transport is TLS [RFC8446].

The Network Configuration Access Control Model (NACM) [RFC8341] provides the means to restrict access for particular NETCONF or RESTCONF users to a preconfigured subset of all available NETCONF or RESTCONF protocol operations and content.

There are a number of data nodes defined in these two YANG modules that are writable/creatable/deletable (i.e., config true, which is the default). These data nodes may be considered sensitive or vulnerable in some network environments. Write operations (e.g., edit-config) and delete operations to these data nodes without proper protection or authentication can have a negative effect on network operations. These are the subtrees and data nodes and their sensitivity/ vulnerability:

'/alto/alto-client/server-discovery': This subtree specifies a set of parameters for an ALTO client to discover ALTO servers. Unauthorized access to it could cause intruders to modify the ALTO discovery parameters (e.g., 'dns-server') in order to expose an ALTO client to fake ALTO servers. Likewise, this data node can be manipulated to prevent an ALTO client from discovering a reachable ALTO server.

'/alto/alto-server/auth-client': This list specifies all the authenticated ALTO clients on an ALTO server. Unauthorized write access to this list can allow intruders to modify the entries so as to add a client that have not been authenticated yet or delete a client that has already been authenticated. Likewise, this data node can be manipulated to prevent access of legitimate ALTO clients.

'/alto/alto-server/role': This list specifies roles which authenticated ALTO clients were assigned to for access control. Unauthorized write access to this list allow intruders to modify the entries so as to permit access that should not be permitted, or deny access that should be permitted.

Some of the readable data nodes in the "ietf-alto" YANG module may be considered sensitive or vulnerable in some network environments. It is thus important to control read access (e.g., via get, get-config, or notification) to these data nodes. These are the subtrees and data nodes and their sensitivity/vulnerability:

'/alto/alto-server/logging-system': This subtree provides configuration to select a logging system to capture log messages generated by an ALTO server. Unauthorized read access of this node can allow intruders to access logging information, which could be used to craft an attack the server.

The "ietf-alto" supports an HTTP listen mode to cover cases where the ALTO server stack does not handle the TLS termination itself, but is handled by a separate component. Special care should be considered when such mode is enabled. Note that the default listen mode is 'https'.

Also, please be aware that these modules include choice nodes that can be augmented by other extended modules. The augmented data nodes may be considered sensitive or vulnerable in some network environments. For instance, an augmented case of the 'source-params' choice in 'data-source' may include authentication information about how to access a data source including private network information. The 'yang-datastore' case in Appendix A.3 is such an example. The 'restconf' and 'netconf' nodes in it may reveal the access to a private YANG datastore. Thus, those extended modules may have the NACM extension "default-deny-all" set.

These modules use groupings defined in other RFCs that define data nodes that do set the NACM "default-deny-all" and "default-deny-write" extensions. Specifically, the following data nodes reuse groupings with their security considerations:

'/alto/alto-server/listen/http/http-server-parameters': This subtree reuses the 'http-server-grouping' grouping defined in [I-D.ietf-netconf-http-client-server]. The security considerations of [I-D.ietf-netconf-http-client-server] have been applied to it. Specifically, the 'server-name' and 'client-authentication' nodes in it may be considered sensitive or vulnerable. For this reason, the NACM extension "default-deny-write" has been applied to them.

'/alto/alto-server/listen/https/http-server-parameters': This subtree reuses the 'http-server-grouping' grouping defined in [I-D.ietf-netconf-http-client-server]. The security considerations of [I-D.ietf-netconf-http-client-server] have been applied to it. Specifically, the 'server-name'and 'client-authentication' nodes in it may be considered sensitive or vulnerable. For this reason, the NACM extension "default-deny-write" has been applied to them.

'/alto/alto-server/listen/https/tls-server-parameters': This subtree reuses the 'tls-server-grouping' grouping defined in [I-D.ietf-netconf-tls-client-server]. The security considerations of [I-D.ietf-netconf-tls-client-server] have been applied to it. Specifically, all the "key" and "private-key" data nodes in it have the NACM extension "default-deny-all" set, thus preventing unrestricted read-access to the cleartext key values. Also, all writable data nodes in it may be considered sensitive or vulnerable. For this reason, the NACM extension "default-deny-write" has been applied to them.

9. IANA Considerations

This document registers the following URIs in the "IETF XML Registry" [RFC3688]:

URI: urn:ietf:params:xml:ns:yang:ietf-alto
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

URI: urn:ietf:params:xml:ns:yang:ietf-alto-stats
Registrant Contact: The IESG.
XML: N/A; the requested URI is an XML namespace.

This document registers the following two YANG modules in the "YANG Module Names" registry [RFC6020]:

Name: ietf-alto
Namespace: urn:ietf:params:xml:ns:yang:ietf-alto
Prefix: alto
Maintained by IANA: N
Reference: [RFC XXXX]

Name: ietf-alto-stats
Namespace: urn:ietf:params:xml:ns:yang:ietf-alto-stats
Prefix: alto-stats
Maintained by IANA: N
Reference: [RFC XXXX]

10. References

10.1. Normative References

[I-D.ietf-netconf-http-client-server]
Watsen, K., "YANG Groupings for HTTP 1.1/2.0 Clients and HTTP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-http-client-server-14, 28 December 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-http-client-server-14>.

[I-D.ietf-netconf-netconf-client-server]
Watsen, K., "NETCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-netconf-client-server-30, 28 December 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-netconf-client-server-30>.

[I-D.ietf-netconf-restconf-client-server]
Watsen, K., "RESTCONF Client and Server Models", Work in Progress, Internet-Draft, draft-ietf-netconf-restconf-client-server-30, 28 December 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-restconf-client-server-30>.

[I-D.ietf-netconf-tcp-client-server]
Watsen, K. and M. Scharf, "YANG Groupings for TCP Clients and TCP Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tcp-client-server-17, 28 December 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tcp-client-server-17>.

[I-D.ietf-netconf-tls-client-server]
Watsen, K., "YANG Groupings for TLS Clients and TLS Servers", Work in Progress, Internet-Draft, draft-ietf-netconf-tls-client-server-34, 28 December 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-tls-client-server-34>.

[I-D.ietf-netconf-trust-anchors]
Watsen, K., "A YANG Data Model for a Truststore", Work in Progress, Internet-Draft, draft-ietf-netconf-trust-anchors-22, 28 December 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-netconf-trust-anchors-22>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, DOI 10.17487/RFC3688, January 2004, <https://www.rfc-editor.org/rfc/rfc3688>.

[RFC5424] Gerhards, R., "The Syslog Protocol", RFC 5424, DOI 10.17487/RFC5424, March 2009, <https://www.rfc-editor.org/rfc/rfc5424>.

[RFC6020] Bjorklund, M., Ed., "YANG - A Data Modeling Language for the Network Configuration Protocol (NETCONF)", RFC 6020, DOI 10.17487/RFC6020, October 2010, <https://www.rfc-editor.org/rfc/rfc6020>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/rfc/rfc6241>.

[RFC6242] Wasserman, M., "Using the NETCONF Protocol over Secure Shell (SSH)", RFC 6242, DOI 10.17487/RFC6242, June 2011, <https://www.rfc-editor.org/rfc/rfc6242>.

[RFC6991] Schoenwaelder, J., Ed., "Common YANG Data Types", RFC 6991, DOI 10.17487/RFC6991, July 2013, <https://www.rfc-editor.org/rfc/rfc6991>.

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <https://www.rfc-editor.org/rfc/rfc7285>.

[RFC7286] Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and H. Song, "Application-Layer Traffic Optimization (ALTO) Server Discovery", RFC 7286, DOI 10.17487/RFC7286, November 2014, <https://www.rfc-editor.org/rfc/rfc7286>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/rfc/rfc8040>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

[RFC8189] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost Application-Layer Traffic Optimization (ALTO)", RFC 8189, DOI 10.17487/RFC8189, October 2017, <https://www.rfc-editor.org/rfc/rfc8189>.

[RFC8341] Bierman, A. and M. Bjorklund, "Network Configuration Access Control Model", STD 91, RFC 8341, DOI 10.17487/RFC8341, March 2018, <https://www.rfc-editor.org/rfc/rfc8341>.

[RFC8342] Bjorklund, M., Schoenwaelder, J., Shafer, P., Watsen, K., and R. Wilton, "Network Management Datastore Architecture (NMDA)", RFC 8342, DOI 10.17487/RFC8342, March 2018, <https://www.rfc-editor.org/rfc/rfc8342>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8686] Kiesel, S. and M. Stiemerling, "Application-Layer Traffic Optimization (ALTO) Cross-Domain Server Discovery", RFC 8686, DOI 10.17487/RFC8686, February 2020, <https://www.rfc-editor.org/rfc/rfc8686>.

[RFC8895] Roome, W. and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Incremental Updates Using Server-Sent Events (SSE)", RFC 8895, DOI 10.17487/RFC8895, November 2020, <https://www.rfc-editor.org/rfc/rfc8895>.

[RFC8896] Randriamasy, S., Yang, R., Wu, Q., Deng, L., and N. Schwan, "Application-Layer Traffic Optimization (ALTO) Cost Calendar", RFC 8896, DOI 10.17487/RFC8896, November 2020, <https://www.rfc-editor.org/rfc/rfc8896>.

[RFC9240] Roome, W., Randriamasy, S., Yang, Y., Zhang, J., and K. Gao, "An Extension for Application-Layer Traffic Optimization (ALTO): Entity Property Maps", RFC 9240, DOI 10.17487/RFC9240, July 2022, <https://www.rfc-editor.org/rfc/rfc9240>.

[RFC9241] Seedorf, J., Yang, Y., Ma, K., Peterson, J., and J. Zhang, "Content Delivery Network Interconnection (CDNI) Footprint and Capabilities Advertisement Using Application-Layer Traffic Optimization (ALTO)", RFC 9241, DOI 10.17487/RFC9241, July 2022, <https://www.rfc-editor.org/rfc/rfc9241>.

[RFC9274] Boucadair, M. and Q. Wu, "A Cost Mode Registry for the Application-Layer Traffic Optimization (ALTO) Protocol", RFC 9274, DOI 10.17487/RFC9274, July 2022, <https://www.rfc-editor.org/rfc/rfc9274>.

[RFC9275] Gao, K., Lee, Y., Randriamasy, S., Yang, Y., and J. Zhang, "An Extension for Application-Layer Traffic Optimization (ALTO): Path Vector", RFC 9275, DOI 10.17487/RFC9275, September 2022, <https://www.rfc-editor.org/rfc/rfc9275>.

[RFC9439] Wu, Q., Yang, Y., Lee, Y., Dhody, D., Randriamasy, S., and L. Contreras, "Application-Layer Traffic Optimization (ALTO) Performance Cost Metrics", RFC 9439, DOI 10.17487/RFC9439, August 2023, <https://www.rfc-editor.org/rfc/rfc9439>.

10.2. Informative References

[I-D.ietf-alto-new-transport]
Gao, K., Schott, R., Yang, Y. R., Delwiche, L., and L. Keller, "The ALTO Transport Information Publication Service", Work in Progress, Internet-Draft, draft-ietf-alto-new-transport-22, 3 January 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-alto-new-transport-22>.

[RFC6291] Andersson, L., van Helvoort, H., Bonica, R., Romascanu, D., and S. Mansfield, "Guidelines for the Use of the "OAM" Acronym in the IETF", BCP 161, RFC 6291, DOI 10.17487/RFC6291, June 2011, <https://www.rfc-editor.org/rfc/rfc6291>.

[RFC7921] Atlas, A., Halpern, J., Hares, S., Ward, D., and T. Nadeau, "An Architecture for the Interface to the Routing System", RFC 7921, DOI 10.17487/RFC7921, June 2016, <https://www.rfc-editor.org/rfc/rfc7921>.

[RFC7971] Stiemerling, M., Kiesel, S., Scharf, M., Seidel, H., and S. Previdi, "Application-Layer Traffic Optimization (ALTO) Deployment Considerations", RFC 7971, DOI 10.17487/RFC7971, October 2016, <https://www.rfc-editor.org/rfc/rfc7971>.

[RFC8340] Bjorklund, M. and L. Berger, Ed., "YANG Tree Diagrams", BCP 215, RFC 8340, DOI 10.17487/RFC8340, March 2018, <https://www.rfc-editor.org/rfc/rfc8340>.

[RFC8346] Clemm, A., Medved, J., Varga, R., Liu, X., Ananthakrishnan, H., and N. Bahadur, "A YANG Data Model for Layer 3 Topologies", RFC 8346, DOI 10.17487/RFC8346, March 2018, <https://www.rfc-editor.org/rfc/rfc8346>.

[RFC8641] Clemm, A. and E. Voit, "Subscription to YANG Notifications for Datastore Updates", RFC 8641, DOI 10.17487/RFC8641, September 2019, <https://www.rfc-editor.org/rfc/rfc8641>.

Appendix A. Examples of Extending the ALTO O&M Data Model

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

* The server-discovery-manner choice of the server-discovery.

* The authentication choice of each auth-client.

* The data-source node.

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

A.1. An Example Module for Extended Server Discovery Manners

The base module "ietf-alto" only includes a reverse DNS based server discovery manner. The following example module demonstrates how additional server discovery methods can be augmented into the base data model.

The case internet-routing-registry allows the ALTO server to update the server URI to the attribute of the corresponding aut-num class in IRR.

The case peeringdb allows the ALTO server to update the server URI to the org object of the organization record in PeeringDB.

```
module example-alto-server-discovery {
  yang-version 1.1;

  namespace "https://example.com/ns/example-alto-server-discovery";
  prefix ex-alto-disc;

  import ietf-alto {
    prefix alto;
    reference
      "RFC XXXX: YANG Data Models for the Application-Layer
                 Traffic Optimization (ALTO) Protocol";
  }

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

  organization
    "Example, Inc.";

  contact
    "Example, Inc.
     Customer Service

     E-mail: alto-oam-yang@example.com";
```

```
  description
    "This module contains a collection of vendor-specific cases of
     server discovery mechanisms for ALTO.";

  revision 2023-02-28 {
    description
      "Version 1.0";
    reference
      "RFC XXXX: YANG Data Models for the Application-Layer
                 Traffic Optimization (ALTO) Protocol";
  }

  augment "/alto:alto/alto:alto-server/alto:server-discovery"
        + "/alto:method" {
    description
      "Examples of server discovery mechanisms provided by the ALTO
       server.";
    case internet-routing-registry {
      description
        "Update descr attributes of an aut-num class in an Internet
         Routing Registry (IRR) database for ALTO server discovery
         using Routing Policy Specification Language (RPSL).";
      reference
        "RFC 2622: Routing Policy Specification Language (RPSL).";
      container irr-params {
        description
          "Configuration parameters for IRR database.";
        leaf aut-num {
          type inet:as-number;
          description
            "The Autonomous System (AS) number to be updated.";
        }
      }
    }
    case peeringdb {
      description
        "Update metadata of a network record in PeeringDB database
         for ALTO server discovery using PeeringDB lookup.";
      container peeringdb-params {
        description
          "Configuration parameters for PeeringDB database.";
        leaf org-id {
          type uint32;
          description
            "Specifies an identifier that refers to the org object
             of the organization record in PeeringDB.";
        }
      }
```

```
      }
    }

    augment "/alto:alto/alto:alto-client"
          + "/alto:server-discovery/alto:method" {
      description
        "Examples of server discovery mechanisms used by an ALTO
         client.";
      case internet-routing-registry {
        description
          "Use IRR to discover an ALTO server.";
        reference
          "RFC 2622: Routing Policy Specification Language (RPSL).";
        container irr-params {
          description
            "Configuration for IRR query using RPSL.";
          leaf whois-server {
            type inet:host;
            description
              "Whois server for an IRR query using RPSL.";
          }
        }
      }
      case peeringdb {
        description
          "Use PeeringDB to discover an ALTO server.";
        container peeringdb-params {
          description
            "Configuration for PeeringDB queries.";
          leaf peeringdb-endpoint {
            type inet:uri;
            description
              "Endpoint of PeeringDB API server.";
          }
        }
      }
    }
  }
```

A.2. An Example Module for Extended Client Authentication Approaches

The base module "ietf-alto" only includes the client authentication approaches directly provided by the HTTP server. However, an implementation may authenticate clients in different ways. For example, an implementation may delegate the authentication to a third-party OAuth 2.0 server. The following example module demonstrates how additional client authentication approaches can enrich the base data model.

In this example, the oauth2 case includes the URI to a third-party OAuth 2.0 based authorization server that the ALTO server can redirect to for the client authentication.

```
module example-alto-auth {
  yang-version 1.1;

  namespace "https://example.com/ns/example-alto-auth";
  prefix ex-alto-auth;

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

  import ietf-alto {
    prefix alto;
    reference
      "RFC XXXX: YANG Data Models for the Application-Layer
                 Traffic Optimization (ALTO) Protocol";
  }

  organization
    "Example, Inc.";

  contact
    "Example, Inc.
     Customer Service

     E-mail: alto-oam-yang@example.com";

  description
    "This module contains a collection of vendor-specific cases of
     client authentication approaches for ALTO.";

  revision 2023-02-28 {
    description
      "Version 1.0";
    reference
      "RFC XXXX: YANG Data Models for the Application-Layer
                 Traffic Optimization (ALTO) Protocol";
  }

  augment "/alto:alto/alto:alto-server/alto:auth-client"
        + "/alto:authentication" {
    description
      "Example of extended ALTO client authentication approaches.";
```

```
      case oauth2 {
        description
          "Example of authentication by a third-party OAuth 2.0
           server.";
        container oauth2 {
          description
            "Parameters for authentication by a third-party OAuth 2.0
             server.";
          leaf oauth2-server {
            type inet:uri;
            description
              "The URI to the authorization server.";
          }
        }
      }
    }
  }
```

A.3. Example Module for Extended Data Sources

The base module "ietf-alto" does not include any choice cases for specific data sources. The following example module demonstrates how a implementation-specific data source can be augmented into the base data model.

The yang-datastore case is used to import the YANG data from a YANG model-driven datastore. It includes:

* datastore to indicate which datastore is fetched.

* target-paths to specify the list of nodes or subtrees in the datastore.

* protocol to indicate which protocol is used to access the datastore. Either restconf or netconf can be used.

```
module example-alto-data-source {
  yang-version 1.1;

  namespace "https://example.com/ns/example-alto-data-source";
  prefix ex-alto-ds;

  import ietf-alto {
    prefix alto;
    reference
      "RFC XXXX: YANG Data Models for the Application-Layer
                 Traffic Optimization (ALTO) Protocol";
  }
```

```
  import ietf-datastores {
    prefix ds;
    reference
      "RFC 8342: Network Management Datastore Architecture (NMDA)";
  }

  import ietf-yang-push {
    prefix yp;
    reference
      "RFC 8641: Subscription to YANG Notifications for Datastore
                 Updates";
  }

  import ietf-netconf-client {
    prefix ncc;
    reference
      "RFC HHHH: NETCONF Client and Server Models";
  }

  import ietf-restconf-client {
    prefix rcc;
    reference
      "RFC IIII: YANG Groupings for RESTCONF Clients and RESTCONF
                 Servers";
  }

  organization
    "Example, Inc.";

  contact
    "Example, Inc.
     Customer Service

     E-mail: alto-oam-yang@example.com";

  description
    "This module contains a collection of vendor-specific cases of
     data sources for ALTO.";

  revision 2023-02-28 {
    description
      "Version 1.0";
    reference
      "RFC XXXX: YANG Data Models for the Application-Layer
                 Traffic Optimization (ALTO) Protocol";
  }

  identity yang-datastore {
```

```
      base alto:source-type;
      description
        "Identity for data source of YANG-based datastore.";
    }

    identity protocol-type {
      description
        "Base identity for protocol type.";
    }

    identity netconf {
      base protocol-type;
      description
        "Identity for NETCONF protocol.";
    }

    identity restconf {
      base protocol-type;
      description
        "Identity for RESTCONF protocol.";
    }

    augment "/alto:alto/alto:alto-server/alto:data-source"
          + "/alto:source-params" {
      when 'derived-from-or-self(../alto:source-type,'
         + '"yang-datastore")';
      description
        "Example data source for local or remote YANG datastore.";
      container yang-datastore-source-params {
        description
          "YANG datastore specific configuration.";
        leaf datastore {
          type ds:datastore-ref;
          mandatory true;
          description
            "Reference of the datastore from which to get data.";
        }
        list target-paths {
          key name;
          description
            "XML Path Language (XPath) to subscribed YANG datastore
             node or subtree.";
          leaf name {
            type string;
            description
              "Name of the supported XPath or subtree filters.";
          }
          uses yp:selection-filter-types;
```

```
        }
        leaf protocol {
          type identityref {
            base protocol-type;
          }
          description
            "Indicates the protocol that is used to access the YANG
             datastore.";
        }
        container restconf {
          uses rcc:restconf-client-app-grouping {
            when 'derived-from-or-self(../protocol, "restconf")';
          }
          description
            "Parameters for the RESTCONF endpoint of the YANG
             datastore.";
        }
        container netconf {
          uses ncc:netconf-client-app-grouping {
            when 'derived-from-or-self(../protocol, "netconf")';
          }
          description
            "Parameters for the NETCONF endpoint of the YANG
             datastore.";
        }
      }
    }
  }
```

A.4. An Example Module for Information Resource Creation Algorithm

The base module "ietf-alto" does not include any choices cases for information resource creation algorithms. But developers may augment the "ietf-alto" module with definitions for 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-alto-alg

  augment /alto:alto/alto:alto-server/alto:resource
            /alto:alto-networkmap-params/alto:algorithm:
    +--:(l3-unicast-cluster)
       +--rw l3-unicast-cluster-algorithm
          +--rw l3-unicast-topo
          |  +--rw source-datastore    alto:data-source-ref
          |  +--rw topo-name?          leafref
          +--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 contains information referring to the target path name of an operational yang-datastore data source node (See Appendix A.3) subscribed in the data-source list (See Section 5.4.1). The referenced target path in the corresponding yang-datastore data source is assumed 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. 'source-datastore' refers to the 'source-id' of the operational yang-datastore data source node, and 'topo-name' refers to the 'name' of the target path in the source datastore.

'depth': This optional parameter sets the depth of the clustering algorithm. For example, if the depth is set 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.

```
module example-alto-alg {
  yang-version 1.1;

  namespace "https://example.com/ns/example-alto-alg";
  prefix ex-alto-alg;

  import ietf-alto {
    prefix alto;
    reference
      "RFC XXXX: YANG Data Models for the Application-Layer
                 Traffic Optimization (ALTO) Protocol";
  }

  import ietf-datastores {
```

```
    prefix ietf-datastores;
    reference
      "RFC 8342: Network Management Datastore Architecture (NMDA)";
  }

  import example-alto-data-source {
    prefix ex-alto-ds;
  }

  organization
    "Example, Inc.";

  contact
    "Example, Inc.
     Customer Service

     E-mail: alto-oam-yang@example.com";

  description
    "This module contains a collection of vendor-specific cases of
     information resource creation algorithms for ALTO.";

  revision 2023-02-28 {
    description
      "Version 1.0";
    reference
      "RFC XXXX: YANG Data Models for the Application-Layer
                 Traffic Optimization (ALTO) Protocol";
  }

  augment "/alto:alto/alto:alto-server/alto:resource"
        + "/alto:alto-networkmap-params/alto:algorithm" {
    description
      "Example of a network map creation algorithm.";
    case l3-unicast-cluster {
      description
        "Example algorithm translating a Layer 3 unicast topology
         of Interface to the Routing System (I2RS) to an ALTO
         network map.";
      container l3-unicast-cluster-algorithm {
        description
          "Parameters for l3-unicast-cluster algorithm.";
        container l3-unicast-topo {
          leaf source-datastore {
            type alto:data-source-ref;
            must 'deref(.)/../alto:source-params'
               + '/ex-alto-ds:yang-datastore-source-params'
               + '/ex-alto-ds:datastore '
```

```
                  + '= "ietf-datastores:operational"'
                  {
                  error-message
                    "The referenced YANG datastore MUST be
                     operational";
                }
                mandatory true;
                description
                  "The data source to YANG datastore.";
              }
              leaf topo-name {
                type leafref {
                  path '/alto:alto/alto:alto-server/alto:data-source'
                     + '[alto:source-id'
                     + ' = current()/../source-datastore]'
                     + '/alto:source-params'
                     + '/ex-alto-ds:yang-datastore-source-params'
                     + '/ex-alto-ds:target-paths/ex-alto-ds:name';
                }
                description
                  "The name of the IETF Layer 3 unicast topology.";
              }
              description
                "The data source info to an IETF Layer 3 unicast
                 topology.";
            }
            leaf depth {
              type uint32;
              description
                "The depth of the clustering.";
            }
          }
        }
      }
    }
```

A.5. Example Usage

This section presents an example showing how the base data model and all the extended models above are used to set up an ALTO server and configure corresponding components (e.g., data source listener, information resource, and access control).

```
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   {
     "ietf-alto:alto": {
       "alto-server": {
         "listen": {
           "https": {
             "tcp-server-parameters": {
               "local-address": "0.0.0.0"
             },
             "alto-server-parameters": {},
             "http-server-parameters": {
               "server-name": "alto.example.com",
               "client-authentication": {
                 "users": {
                   "user": [
                     {
                       "user-id": "alice",
                       "basic": {
                         "user-id": "alice",
                         "password": "$0$p8ssw0rd"
                       }
                     }
                   ]
                 }
               }
             },
             "tls-server-parameters": {
               "server-identity": {}
             }
           }
         },
         "server-discovery": {
           "example-alto-server-discovery:irr-params": {
             "aut-num": 64496
           }
         },
         "auth-client": [
           {
             "client-id": "alice",
             "https-auth-client": {
               "user-id": "alice"
             }
           },
           {
             "client-id": "bob",
             "example-alto-auth:oauth2": {
               "oauth2-server": "https://auth.example.com/login"
```

```
            }
          }
        ],
        "role": [
          {
            "role-name": "group0",
            "client": [
              "alice",
              "bob"
            ]
          }
        ],
        "data-source": [
          {
            "source-id": "test-yang-ds",
            "source-type": "example-alto-data-source:yang-datastore",
            "source-params": {
              "example-alto-data-source:yang-datastore-source-params"\
                                                                    : {
                "datastore": "ietf-datastores:operational",
                "target-paths": [
                  {
                    "name": "network-topology",
                    "datastore-xpath-filter": "/network-topology:\
    network-topology/topology[topology-id=bgp-example-ipv4-topology]"
                  }
                ],
                "protocol": "restconf",
                "restconf": {
                  "listen": {
                    "endpoint": [
                      {
                        "name": "example restconf server",
                        "https": {
                          "tcp-server-parameters": {
                            "local-address": "192.0.2.2"
                          },
                          "http-client-parameters": {
                            "client-identity": {
                              "basic": {
                                "user-id": "carol",
                                "cleartext-password": "secret"
                              }
                            }
                          }
                        }
                      }
                    ]
```

```
                    }
                  }
                }
              }
            }
          ],
          "resource": [
            {
              "resource-id": "default-network-map",
              "resource-type": "network-map",
              "accepted-role": [
                "group0"
              ],
              "alto-networkmap-params": {
                "is-default": true,
                "example-alto-alg:l3-unicast-cluster-algorithm": {
                  "l3-unicast-topo": {
                    "source-datastore": "test-yang-ds",
                    "topo-name": "network-topology"
                  },
                  "depth": 2
                }
              }
            }
          ]
        }
      }
    }
```

Note that this example only uses a clear text password for demonstration purpose. In practice, it is NOT RECOMMENDED to use any clear text passwords when using this data model.

Appendix B. A Sample ALTO Server Architecture to Implement ALTO O&M YANG Modules

Figure 11 shows a sample architecture for an ALTO server implementation. It indicates the major server components that an ALTO server usually needs to include and the YANG modules that these server components need to implement.

> This section does not intend to impose an internal structure of server implementations, but is provided to exemplify how the various data model components can be used, including having provisions for future augmentations.

The following server components need to implement the 'ietf-alto' module (Section 5):

Server manager: Provides the functionality and configuration of the server-level management, including server listen stack setup, server discovery setup, logging system configuration, global metadata, and server-level security configuration.

Information resource manager: Provides the operation and management for creating, updating, and removing ALTO information resources.

Data source listener: An ALTO server may start multiple data source listeners. Each data source listener defines a communication endpoint that can fetch ALTO-related information from data sources. The information can be either raw network/computation information or pre-processed ALTO-level information.

The following components need to implement the 'ietf-alto-stats' module (Section 6) to provide statistics information:

Performance monitor: Collects ALTO-specific performance metrics at a running ALTO server.

Logging and fault manager: Collects runtime logs and failure events that are generated by an ALTO server and the service of each ALTO information resource.

The following components are also important for an ALTO server, although they are not in the scope of the data models defined in this document:

Data broker: An ALTO server may implement a data broker to store network/computation information collected from data sources or cache some preprocessed data. The service of the ALTO information resource can read them from the data broker to calculate ALTO responses and return to ALTO clients.

Algorithm plugin: The service of each ALTO information resource needs to configure an algorithm to decide how to calculate the ALTO responses. The algorithm plugins implement those algorithms. User-specified YANG modules can be applied to different algorithm plugins by augmenting the ALTO modules (Appendix A).

Generally, the ALTO server components illustrated above have the following interactions with each other:

* Both the server manager and information resource manager will report statistics data to the performance monitor and the logging and fault manager.

* The algorithm plugins will register callbacks to the corresponding ALTO information resources upon configuration; Once an ALTO information resource is requested, the registered callback algorithm will be invoked.

* A data source listener will fetch data from the configured data source using the corresponding data source API in either proactive mode (polling) or reactive mode (subscription/publication).

* A data source listener will send the preprocessed data to a data broker.

* An algorithm plugin may read data from an optional data broker to calculate the ALTO information resource.

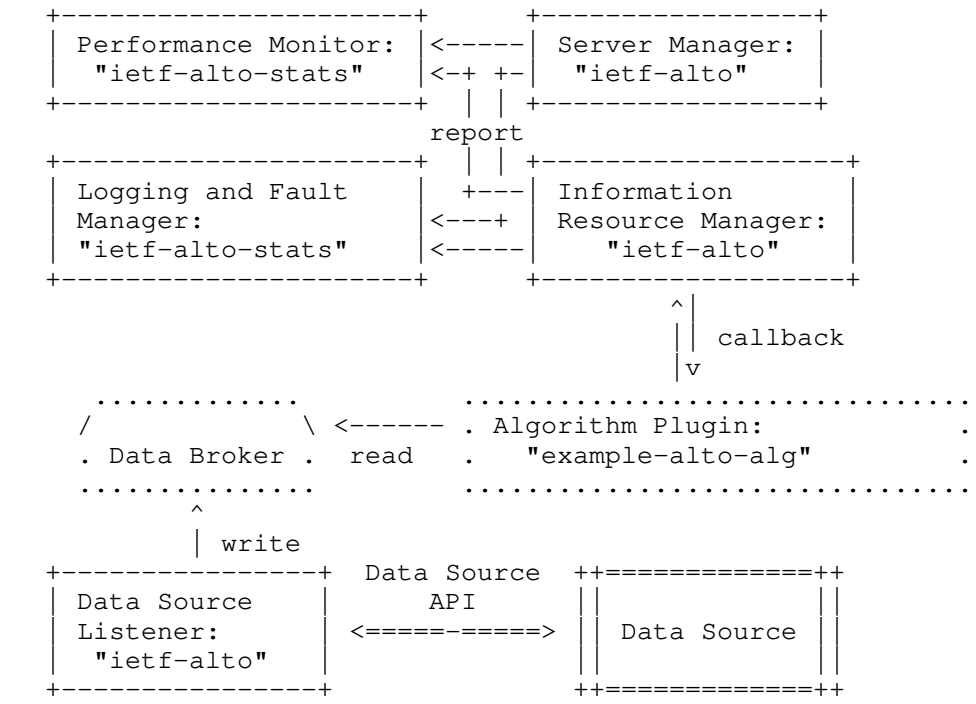


Figure 11: A Sample ALTO Server Architecture and ALTO YANG Modules

Acknowledgements

The authors thank Qin Wu for the help with drafting the initial version of the YANG modules. Big thanks also to ALTO WG chairs (Mohamed Boucadair and Qin Wu) and Adrian Farrel, Dong Guo, Jordi Ros Giralt, Luis M. Contreras, Mahdi Soleimani, Qiao Xiang, Shenshen Chen, and Y. Richard Yang for their thorough reviews, shepherding, and valuable feedback.

Thanks to Dan Romascanu for the opsdir and genart reviews, Andy Bierman for the yangdoctors review, Spencer Dawkins for the tsvart review, Scott Rose for the dnsdir review, and Rich Salz for the secdir review.

Thanks to Martin Duke for the careful AD review.

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
Sichuan, 610000
China
Email: kaigao@scu.edu.cn

Roland Schott
Deutsche Telekom
Ida-Rhodes-Straße 2
64295 Darmstadt
Germany
Email: Roland.Schott@telekom.de

Qiufang Ma
Huawei
101 Software Avenue, Yuhua District
Nanjing
Jiangsu, 210012
China
Email: maqiufang1@huawei.com

ALTO
Internet-Draft
Intended status: Informational
Expires: January 12, 2023

L. Contreras
Telefonica
S. Randriamasy
Nokia Bell Labs
X. Liu
IBM Corporation

July 11, 2022

ALTO extensions for handling Service Functions
draft-lcsr-alto-service-functions-01

Abstract

This document proposes the usage of ALTO (and its extensions) to provide information about service functions to clients (e.g., external systems) that could consume such information for decisions requiring network information (service composition, traffic steering to service chains, etc).

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 January 12, 2023.

Table of Contents

1. Introduction

Network services are commonly formed by means of the concatenation of several atomic service functions (SF), resulting in a connected graph of functions. Those functions can be topologically spread across the network. In addition to that, there will be typically more than one instance of any particular atomic service function in the network for different purposes such as load balancing, redundancy, traffic optimization, etc.

During the definition phase of a network service there will be a process for defining the type of service functions needed for implementing a given network service, as well as the way in which they should be connected to steer the traffic flows through them. The type of a SF can be for instance a User Plane Function (UPF) of the mobile packet core, a cache of a Content Delivery Network (CDN), etc. Thus when having multiple instances of a function (i.e., multiple UPFs or multiple caches as in the example before), a decision process should be in place to determine the particular instances for each type of service function (i.e., what instance of UPF and CDN cache) to be part of the realization of the network service.

At this point of network service realization, having timely information of the characteristics of the interconnection paths among SFs can be crucial. Aspects such as number of hops, associated performance metrics, etc., can enrich (or even determine) the decision of which instances of the service function consider as final election.

This document proposes the usage of ALTO [RFC7285] and its extensions to provide information about service functions or their interconnection paths to clients (e.g., external systems) that could consume such information for decisions requiring network information.

2. Service Function information

Several initiatives in IETF deal with the interconnection of service functions.

[RFC7665] defines the Service Function Chain (SFC) architecture. There, the traffic is steered through SFC domains with the objective of making the flow passing through a number of service functions to run a service. When entering the domain, the traffic is classified and assigned to an SFC Path. Specific information is added to the packet flows within the domain, being this SFC encapsulation containing metadata and contextual information useful for the processing of the flows by the service functions and other components in the architecture.
In all this process, there is no explicit identification of the service function to direct the traffic to, as it is implicit in the definition of a specific SFC Path.

Similarly, in [RFC8986] the Segment Identifiers of SRv6 structures the 128 bits of the IPv6 address in the form LOC:FUNCT:ARG. LOC is the locator used to route a packet to the endpoint and encoded in the L most significant bits of the Segment Identifier (SID) .FUNCT represents a Function ID and uses F bits. ARG represents optional parameters to be interpreted by the function, and uses A bits. Furthermore, [I-D.ietf-spring-sr-service-programming] defines data plane functionalities required to implement service segments, in a similar way as [RFC7665] for SFC.

Finally, [I-D.ietf-teas-sf-aware-topo-model] proposes a YANG data model able to integrate both network topology and service location on the same traffic engineering topology. In this model, the service functions are represented by service-function-id and sf-connection-point-id.

In all these previous cases, the information relative to the service functions is quite limited, if present. Richer information could be needed for an integration between the control systems responsible for the service operation and the control systems responsible for the network actions that could optimize the delivery of services relying on network information (that is, acting in an integrated fashion).

For instance, taking as example OpenStack [OpenStack], a network service relies on descriptors providing information about Virtual

Deployment Units (VDUs), Connection Points (CPs) and Virtual Links (VLs).

A VDU describes the properties of the virtual construct that hosts the service function. Important information is the function identifier and its type. The CPs contain the IP and MAC addresses for such function, showing the binding as well between a VDU and a VL. Finally, the VL identifies the connectivity between VDUs.

The level of information is not the same in all the solutions overviewed, however a solution like ALTO could help to reconcile all these different approaches by mapping and matching information on the service and the network planes.

3. Usage of ALTO for retrieving information relative to service functions

ALTO can expose combined information of the service overlay together with the underlying network characteristics. This section details the potential usage of ALTO in this respect.

3.1. Information of interest

There can be several kinds of ALTO information requests to take into consideration. Some examples are listed below:

- Path characteristics, from a PID, to any instance of a service function type.
- Path characteristics, from a PID, to a specific instance of a service function type.
- Path characteristics among any instance of a service function type X to any other instance of a service function type Y.
- Path characteristics among a specific instance of a service function type X to any other instance of a service function type Y.
- Path characteristics, from a PID, to a chain of service functions.
- Path characteristics, from a PID, to a chain of specific instances of service functions.

Other type of requests could be further identified.

An ALTO server could be able to provide information for a limited set of requests. Thus, some indication of the possible requests to be served should be in place when interacting with the client.

3.2. ALTO mechanisms to support the requests about service functions

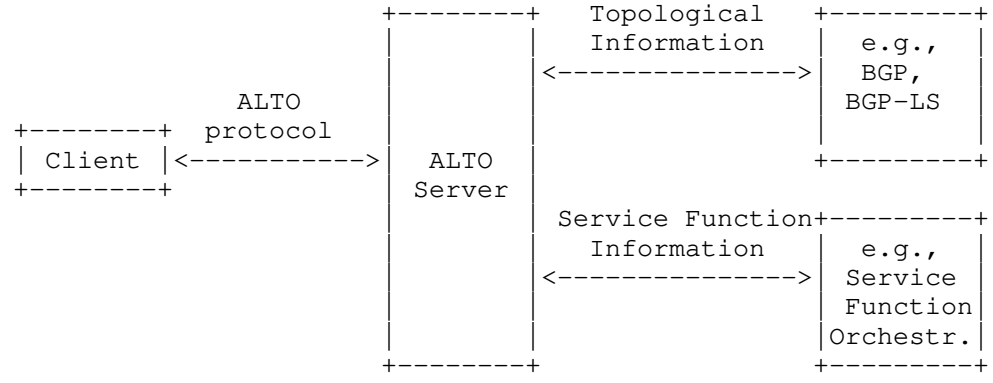
ALTO can determine the path characteristics between two endpoints as determined by [RFC7285]. ALTO also can provide the view of chain of functions by leveraging on the path vector concept developed in [I-D.ietf-alto-path-vector], where the endpoints considered represent service functions.

[I-D.ietf-alto-path-vector] introduces the concept of Abstract Network Element (ANE) to specify a component or an aggregation of components sharing some characteristics in a network. Furthermore, [I-D.ietf-alto-unified-props-new] generalizes the concept of endpoint properties to entity properties, where entities may be defined in semantic domains such as as IPv4 or IPv6, or PIDs or ANEs.

This draft makes use of these capabilities to support the retrieval of information relative to service functions.

4. ALTO architecture for service function information retrieval

The following logical architecture defines the usage of ALTO for the retrieval of information about service functions or interconnection of service functions.



The network topological information will be complemented with information relative to the service functions as provided by the orchestration system managing and controlling that part.

The ALTO server will integrate the information of the service functions based on some parameters, such as the IP address of the service functions.

5. Proposed ALTO extensions

As proposed extension to existing ALTO specifications, the following aspects are considered:

- Extension to ALTO protocol to allow ALTO clients to express detailed requests in line with the information of interest described in Section 3.1.

- Extensions to ALTO in order to collect and combine both service and network information, in line with the architecture depicted in Section 3.3. These extensions can involve particularizations of both [I-D.ietf-alto-path-vector] and [I-D.ietf-alto-unified-props-new].

Further extensions could be required.

Next iterations of this draft will further analyze the gap between existing ALTO features and requirements to support the provisioning of infrastructure information needed to perform efficient SF management.

6. Security Considerations

To be provided.

7. Informative References

[I-D.ietf-alto-path-vector]
Gao, K., Lee, Y., Randriamasy, S., Yang, Y. R., and J. J. Zhang, "An ALTO Extension: Path Vector", draft-ietf-alto-path-vector-25 (work in progress), March 2022.

[I-D.ietf-alto-unified-props-new]
Roome, W., Randriamasy, S., Yang, Y. R., Zhang, J. J., and K. Gao, "An ALTO Extension: Entity Property Maps", draft-ietf-alto-unified-props-new-24 (work in progress), February 2022.

[I-D.ietf-spring-sr-service-programming]
Clad, F., Xu, X., Filsfils, C., Bernier, D., Li, C., Decraene, B., Ma, S., Yadlapalli, C., Henderickx, W., and S. Salsano, "Service Programming with Segment Routing", draft-ietf-spring-sr-service-programming-06 (work in progress), June 2022.

[I-D.ietf-teas-sf-aware-topo-model]
Bryskin, I., Liu, X., Lee, Y., Guichard, J., Contreras, L. M., Ceccarelli, D., Tantsura, J., and D. Shytyi, "SF Aware TE Topology YANG Model", draft-ietf-teas-sf-aware-topo-model-09 (work in progress), February 2022.

[OpenStack]
"VNF Descriptor (VNFD) Template Guide, https://docs.openstack.org/tacker/latest/contributor/vnfd_template_description.html", n.d..

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <https://www.rfc-editor.org/info/rfc7285>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

Authors' Addresses

Luis M. Contreras
Telefonica

Email: luismiguel.contrerasmurillo@telefonica.com

Sabine Randriamasy
Nokia Bell Labs

Email: sabine.randriamasy@nokia-bell-labs.com

Xufeng Liu
IBM Corporation

Email: xufeng.liu.ietf@gmail.com

ALTO
Internet-Draft
Intended status: Informational
Expires: 14 September 2023

L. M. Contreras, Ed.
Telefonica
S. Randriamasy
Nokia Bell Labs
X. Liu
IBM Corporation
13 March 2023

ALTO extensions for handling Service Functions
draft-lcsr-alto-service-functions-02

Abstract

This document proposes the usage of ALTO (and its extensions) to provide information about service functions to clients (e.g., external systems) that could consume such information for decisions requiring network information (service composition, traffic steering to service chains, etc).

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 14 September 2023.

Copyright Notice

Table of Contents

1. Introduction

Network services are commonly formed by means of the concatenation of several atomic service functions (SF), resulting in a connected graph of functions. Those functions can be topologically spread across the network. In addition to that, there will be typically more than one instance of any particular atomic service function in the network for different purposes such as load balancing, redundancy, traffic optimization, etc.

During the definition phase of a network service there will be a process for defining the type of service functions needed to implement a given network service, as well as the way in which they should be connected to steer the traffic flows through them. The type of a SF can be for instance a User Plane Function (UPF) of the mobile packet core, a cache of a Content Delivery Network (CDN), etc. Thus when having multiple instances of a function (i.e., multiple UPFs or multiple caches as in the example before), a decision process should be in place to determine the particular instances for each type of service function (i.e., what instance of UPF and CDN cache) that will be part of the realization of the network service, that we refer to as network service instance in this document.

Information on the SFs and their instances is available from entities such as Service Orchestrators. Examples of such information are SF type, service instance ID, SF locator, etc. Information about a network service instance can be more comprehensively defined in terms of performace and resources efficiency if this information is complemented with network capabilities. This draft proposes to complement information available on SF or their instances with information on the paths that interconnect them.

At this point of network service realization, having timely information of the characteristics of the interconnection paths among SFs can be crucial. Aspects such as number of hops, associated performance metrics, etc., can enrich (or even determine) the decision of which instances of the service function consider as final election.

This document proposes the usage of ALTO [RFC7285] and its extensions to provide information about service functions or their interconnection paths to clients (e.g., external systems) that could consume such information for decisions requiring network information.

2. Service Function information

2.1. Cases relevant for exposing information related to Service Functions

Several initiatives in IETF deal with the interconnection of service functions.

[RFC7665] defines the Service Function Chain (SFC) architecture. There, the traffic is steered through SFC domains with the objective of making the flow passing through a number of service functions to run a service. When entering the domain, the traffic is classified and assigned to an SFC Path. Specific information is added to the packet flows within the domain, being this SFC encapsulation

containing metadata and contextual information useful for the processing of the flows by the service functions and other components in the architecture. In all this process, there is no explicit identification of the service function to direct the traffic to, as it is implicit in the definition of a specific SFC Path.

Similarly, in [RFC8986] the Segment Identifiers of SRv6 structures the 128 bits of the IPv6 address in the form LOC:FUNCT:ARG. LOC is the locator used to route a packet to the endpoint and encoded in the L most significant bits of the Segment Identifier (SID). FUNCT represents a Function ID and uses F bits. ARG represents optional parameters to be interpreted by the function, and uses A bits. Furthermore, [I-D.ietf-spring-sr-service-programming] defines data plane functionalities required to implement service segments, in a similar way as [RFC7665] for SFC.

Moreover, [I-D.ietf-teas-sf-aware-topo-model] defines a YANG data model able to integrate both network topology and service location on the same traffic engineering topology. In this model, the service functions are represented by service-function-id and sf-connection-point-id.

Finally, [I-D.ldbc-cats-framework] proposes a framework in which the metrics associated to different service instances of the same service function type are used for taking traffic steering decisions using overlay connections. The metrics considered in such decision are expected to be a combination of networking and computing metrics.

In all these previous cases, the information relative to the service functions can be limited, if present. Richer information could be needed for an integration between the control systems responsible for the service operation and the control systems responsible for the network actions that could optimize the delivery of services relying on network information (that is, acting in an integrated fashion).

For instance, taking as example OpenStack [OpenStack], a network service relies on descriptors providing information about Virtual Deployment Units (VDUs), Connection Points (CPs) and Virtual Links (VLs).

A VDU describes the properties of the virtual construct that hosts the service function. Important information is the function identifier and its type. The CPs contain the IP and MAC addresses for such function, showing the binding as well between a VDU and a VL. Finally, the VL identifies the connectivity between VDUs.

The level of information is not the same in all the solutions overviewed, however a solution like ALTO could help to reconcile all these different approaches by mapping and matching information on the service and the network planes.

2.2. Retrieval of Service Function information

Different options can be considered, in some cases being complementary and in some other cases being actual alternatives.

One option is the retrieval of information as provided by Service Function Orchestration systems, as described by [OpenStack] or similar solutions.

In addition to that, or as an alternative, different routing protocol extensions can provide information about existing Service Functions in the network. For instance [RFC9015] can advertise using BGP the Service Function Type and Service Function Instances in a network. Furthermore, [I-D.xu-lsr-isis-service-function-adv] and [I-D.xu-lsr-ospf-service-function-adv] can advertise at IGP level the Service Function Identifier as unique identifier representing a service function within a domain.

3. Usage of ALTO for retrieving information relative to service functions

ALTO can expose combined information of the service overlay together with the underlying network characteristics. This section details the potential usage of ALTO in this respect.

3.1. Information of interest

There can be several kinds of ALTO exposure information in response to client requests that can be taken into consideration. Some examples are listed below:

* Path characteristics, from a PID, to any instance of a service function type.

* Path characteristics, from a PID, to a specific instance of a service function type.

* Path characteristics among any instance of a service function type X to any other instance of a service function type Y.

* Path characteristics among a specific instance of a service function type X to any other instance of a service function type Y.

* Path characteristics, from a PID, to a chain of service functions.

* Path characteristics, from a PID, to a chain of specific instances of service functions.

Other type of requests could be further identified. The information to be exposed can be a combination of network and service function related characteristics.

An ALTO server could be able to provide information for a limited set of requests. Thus, some indication of the possible requests to be served should be in place when interacting with the client.

3.2. ALTO mechanisms to support the requests about service functions

ALTO can determine the path characteristics between two endpoints as described in [RFC7285]. ALTO also can provide the view of chain of functions by leveraging on the path vector concept developed in [RFC9275], where the endpoints considered represent service functions.

[RFC9275] introduces the concept of Abstract Network Element (ANE) to specify a component or an aggregation of components sharing some characteristics in a network. Furthermore, [RFC9240] generalizes the concept of endpoint properties to entity properties, where entities may be defined in semantic domains such as as IPv4 or IPv6, or PIDs or ANEs.

This draft makes use of these capabilities to support the retrieval of information relative to service functions.

4. ALTO architecture for service function information retrieval

The following logical architecture defines the usage of ALTO for the retrieval of information about service functions or interconnection of service functions.

Figure 1 represents the information exchange of an ALTO Server for providing information about service functions to ALTO clients (e.g., external systems) that require combined information of network and service functions.

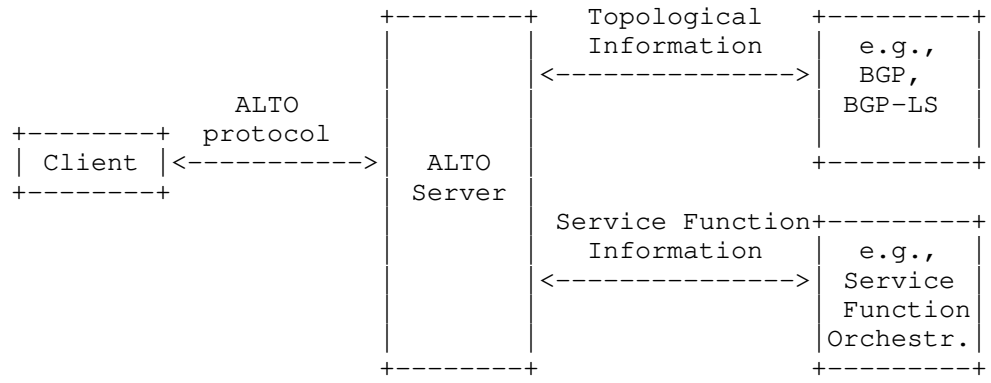


Figure 1. Information exchange

The network topological information will be complemented with information relative to the service functions as provided by the orchestration system managing and controlling that part.

The ALTO server will integrate the information of the service functions based on some parameters, such as the IP address of the service functions.

5. Proposed ALTO extensions

As proposed extension to existing ALTO specifications, the following aspects are considered:

* Extension to ALTO protocol to allow ALTO clients to express detailed requests in line with the information of interest described in Section 3.1.

* Extensions to ALTO in order to combine and expose both service and network information, in line with the architecture depicted in Section 3.3. These extensions can involve particularizations of both [RFC9275] and [RFC9240].

* Identification of mechanisms to be used by ALTO for collecting service functions' information.

Further extensions could be required.

Next iterations of this draft will further analyze the gap between existing ALTO features and requirements to support the provisioning of infrastructure information needed to perform efficient SF management.

6. Security Considerations

To be provided.

7. Informative References

[I-D.ietf-spring-sr-service-programming]
Clad, F., Xu, X., Filsfils, C., Bernier, D., Li, C., Decraene, B., Ma, S., Yadlapalli, C., Henderickx, W., and S. Salsano, "Service Programming with Segment Routing", Work in Progress, Internet-Draft, draft-ietf-spring-sr-service-programming-07, 15 February 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-spring-sr-service-programming-07>.

[I-D.ietf-teas-sf-aware-topo-model]
Bryskin, I., Liu, X., Lee, Y., Guichard, J., Contreras, L. M., Ceccarelli, D., Tantsura, J., and D. Shytyi, "SF Aware TE Topology YANG Model", Work in Progress, Internet-Draft, draft-ietf-teas-sf-aware-topo-model-11, 12 March 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-teas-sf-aware-topo-model-11>.

[I-D.ldbc-cats-framework]
Li, C., Du, Z., Boucadair, M., Contreras, L. M., Drake, J., Huang, D., and G. S. Mishra, "A Framework for Computing-Aware Traffic Steering (CATS)", Work in Progress, Internet-Draft, draft-ldbc-cats-framework-01, 10 March 2023, <https://datatracker.ietf.org/doc/html/draft-ldbc-cats-framework-01>.

[I-D.xu-lsr-isis-service-function-adv]
Xu, X., Huang, H., Shah, H. C., and L. M. Contreras, "Advertising Service Functions Using IS-IS", Work in Progress, Internet-Draft, draft-xu-lsr-isis-service-function-adv-00, 9 March 2023, <https://datatracker.ietf.org/doc/html/draft-xu-lsr-isis-service-function-adv-00>.

[I-D.xu-lsr-ospf-service-function-adv]
Xu, X., Huang, H., Shah, H. C., and L. M. Contreras, "Advertising Service Functions Using OSPF", Work in Progress, Internet-Draft, draft-xu-lsr-ospf-service-function-adv-00, 9 March 2023, <https://datatracker.ietf.org/doc/html/draft-xu-lsr-ospf-service-function-adv-00>.

[OpenStack] "VNF Descriptor (VNFD) Template Guide, https://docs.openstack.org/tacker/latest/contributor/vnfd_template_description.html", n.d..

[RFC7285] Alimi, R., Ed., Penno, R., Ed., Yang, Y., Ed., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, DOI 10.17487/RFC7285, September 2014, <https://www.rfc-editor.org/info/rfc7285>.

[RFC7665] Halpern, J., Ed. and C. Pignataro, Ed., "Service Function Chaining (SFC) Architecture", RFC 7665, DOI 10.17487/RFC7665, October 2015, <https://www.rfc-editor.org/info/rfc7665>.

[RFC8986] Filsfils, C., Ed., Camarillo, P., Ed., Leddy, J., Voyer, D., Matsushima, S., and Z. Li, "Segment Routing over IPv6 (SRv6) Network Programming", RFC 8986, DOI 10.17487/RFC8986, February 2021, <https://www.rfc-editor.org/info/rfc8986>.

[RFC9015] Farrel, A., Drake, J., Rosen, E., Uttaro, J., and L. Jalil, "BGP Control Plane for the Network Service Header in Service Function Chaining", RFC 9015, DOI 10.17487/RFC9015, June 2021, <https://www.rfc-editor.org/info/rfc9015>.

[RFC9240] Roome, W., Randriamasy, S., Yang, Y., Zhang, J., and K. Gao, "An Extension for Application-Layer Traffic Optimization (ALTO): Entity Property Maps", RFC 9240, DOI 10.17487/RFC9240, July 2022, <https://www.rfc-editor.org/info/rfc9240>.

[RFC9275] Gao, K., Lee, Y., Randriamasy, S., Yang, Y., and J. Zhang, "An Extension for Application-Layer Traffic Optimization (ALTO): Path Vector", RFC 9275, DOI 10.17487/RFC9275, September 2022, <https://www.rfc-editor.org/info/rfc9275>.

Authors' Addresses

Luis M. Contreras (editor)
Telefonica
Ronda de la Comunicacion, s/n
28050 Madrid
Spain
Email: luismiguel.contrerasmurillo@telefonica.com
URI: http://lmcontreras.com

Sabine Randriamasy
Nokia Bell Labs
Email: sabine.randriamasy@nokia-bell-labs.com

Xufeng Liu
IBM Corporation
Email: xufeng.liu.ietf@gmail.com

alto Working Group ZJ. Sun, H.Yang, C.Li
Internet-Draft Beijing University of Posts and Telecommunications
Intended status: Standards Track S.Liu
Expires: 11 January 2023 China Mobile Research Istitute
H. Zheng
Huawei Technologies
25 July 2023

Architecture of Computing Power Optical Network
draft-sun-alto-arch-computing-optical-network-03

Abstract

This document describes the architecture of computing power optical network.

Table of Contents

1. Introduction

With the rapid popularization and application of cloud computing, artificial intelligence and other technologies, the total amount of data has increased explosively, and the demand for data storage, computing and transmission has increased significantly. This puts forward higher requirements for flexible network scheduling and quality of service. More importantly, the upgrading of industrial intelligence will bring about the diversity of devices, such as the application of Internet of things (IOT) sensors, cameras and other devices will produce diverse data. The processing of these heterogeneous data needs ubiquitous computing power to support.

The computing power network is the link that efficiently connects ubiquitous computing power resources and massive user data. With the advantages of ultra-large capacity, ultra-long distance, low latency, and flexible scheduling, optical networks provide a wide coverage, flexible and efficient super-capacity guarantee for computing resources.

The architecture of Computing Power Optical Network supports network-aware applications, networks, computing power and user needs, coordinates the scheduling of computing power resources and network resources, and provides the best user experience. This architecture combines the computing power network with the optical network to realize the collaborative linkage between edge computing and cloud computing.

1.1. 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.

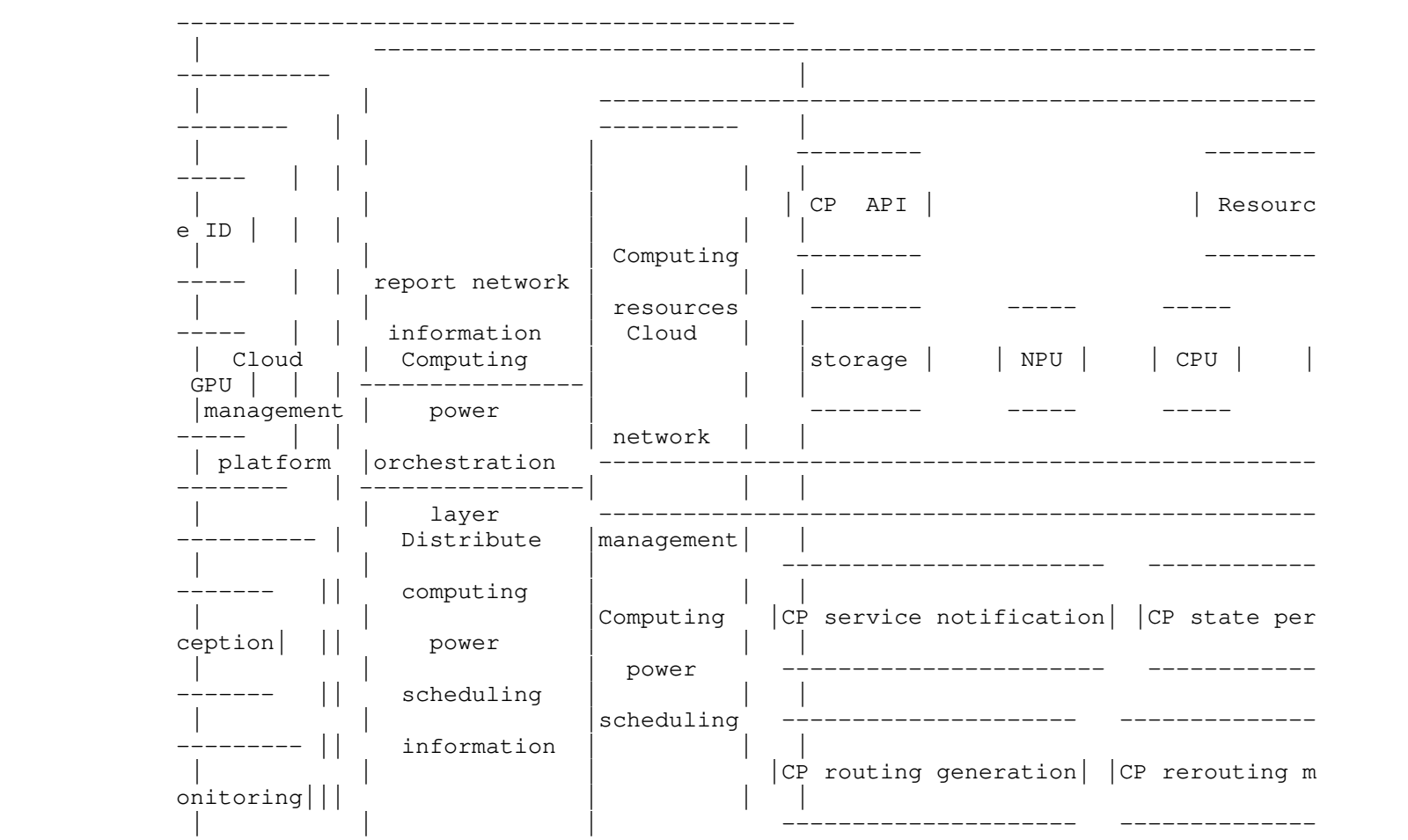
2. Scenarios

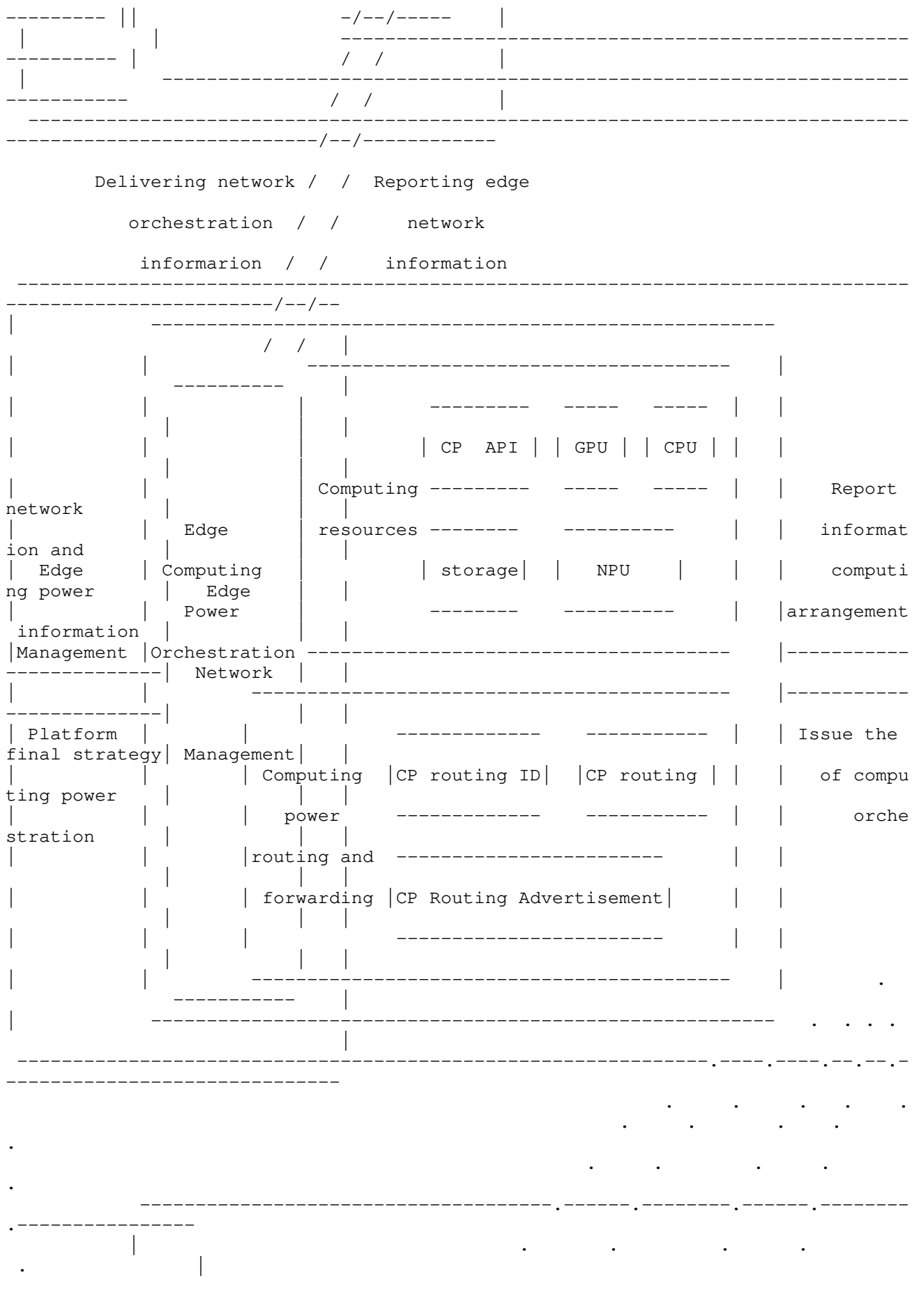
With the prevalence of cloud services, enterprise services and other services, the architecture of computing power optical network has become the choice to solve supported services. The following scenarios provide some typical applications.

2.1. Network Resource Acquirement

The edge network management layer receives information from the client, obtains complete user information, and provides it to the cloud management platform for network resource synchronization. The cloud management platform obtains regular information about applications and networks.

3. The architecture of Computing Power Optical Network




Delivering network orchestration informarion
Reporting edge network information
CP API
GPU
CPU
Computing resources
storage
NPU
Edge Computing Power
Network Management Orchestration
Report network information and Edge Computing power information
Platform Management
Computing power routing and forwarding
CP routing ID
CP routing
CP Routing Advertisement
Issue the final strategy of computing power orchestration

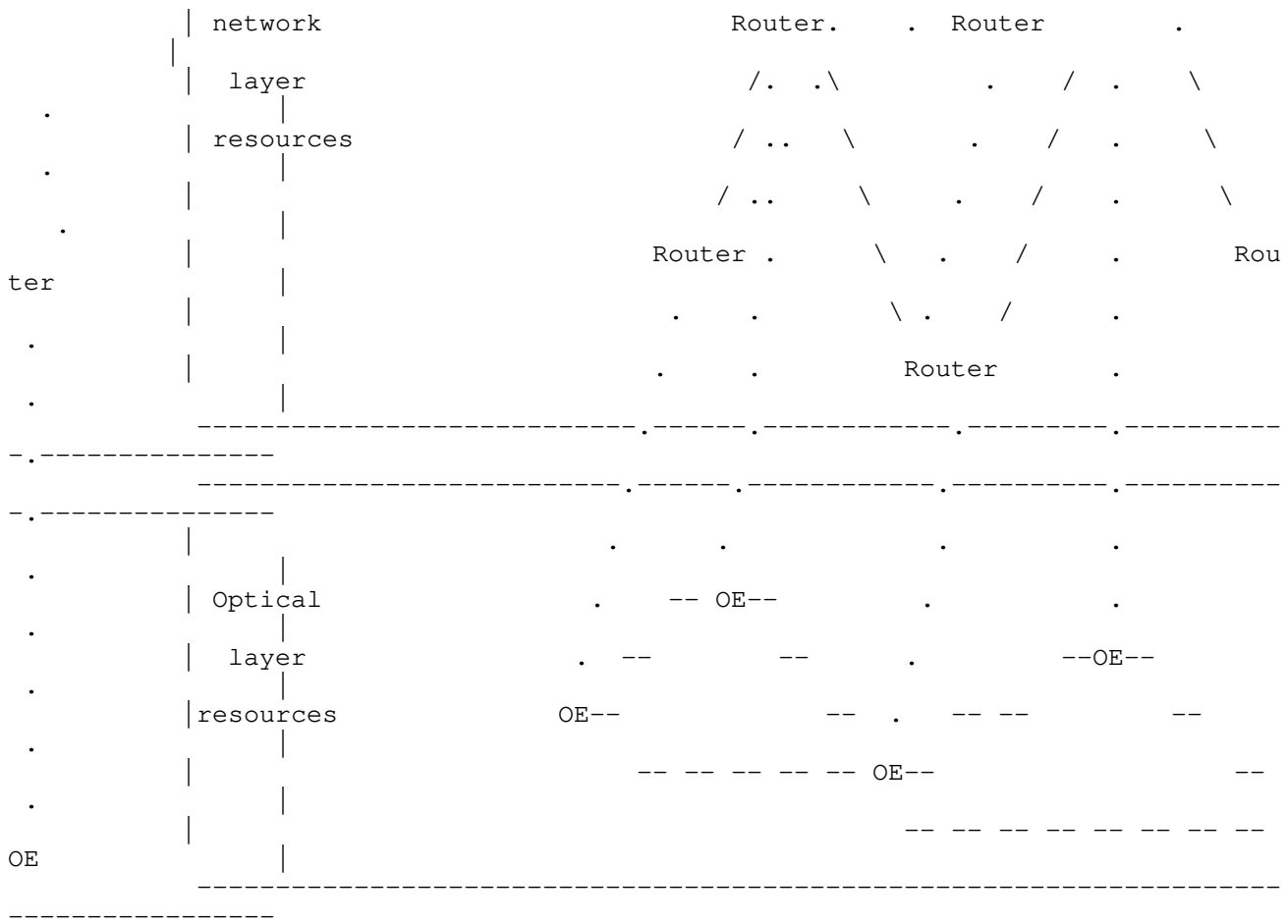


Fig.1 The architecture of computing power optical network.

3.1. Cloud management platform

In order to realize the perception of ubiquitous computing and services, cloud management platform includes computing power scheduling and network management. Computing power scheduling can be divided into computing power resources and computing power scheduling. Computing resources are the use of existing computing infrastructure to provide computing resources. The computing infrastructure is mainly composed of edge computing nodes and network devices, which are controlled by the computing network control layer through the north interface, and provide computing, storage and network facility resources for the serverless edge computing network reference architecture. The computing infrastructure includes a combination of various computing capabilities such as single core central processing unit (CPU), graphics processor (GPU), network processor (NPU). In order to meet the diverse computing needs of the edge computing field, this layer can provide functions such as algorithm library, computing application programming interface (API), computing network resource identification, etc.

The computing power scheduling layer is the core of the computing power aware network, which consists of computing power service notification, computing power state awareness, computing power route generation and computing power route monitoring. Based on the abstracted computing network resources, and considering the network status and computing resource status comprehensively, the computing

power routes that can flexibly schedule services to different computing resource nodes on demand are generated, and the real-time monitoring of computing power routes is carried out. Computing power nodes, including terminals, edges and cloud data centers, need to collect and distribute information about their computing power resource status, such as CPU processing capacity, queue status, cache status, computing power node address, etc. through the control surface network mechanism, that is, create a global computing power routing table at the forwarding node of the whole network computing power network, in case of application requests for optimal routing scheduling in the whole network computing resource pool. The cloud network management layer reports the current information of the network to the computing force arrangement layer and accepts the computing force arrangement information issued by the computing force arrangement layer.

3.2. Edge management platform

Edge management platform includes edge computing force arrangement and edge network management. Edge computing scheduling can be divided into computing resources and computing routing forwarding. Computing resources include computing application programming interface (API), central processing unit (CPU), graphics processor (GPU), network processor (NPU), and storage composition. They are controlled by the computing network control layer throug h the north interface, providing computing, storage and other resources for the server free edge computing network reference architecture.

The computing power route forwarding layer is composed of computing power route identification, computing power route addressing, computing power route notific ation .
Through the distributed edge computing nodes, through the automatic deployment of services, optimal routing and cross layer optimization, the edge computing powe r aware network is built, which can truly call different computing resources on demand and in real time, improve the utilization efficiency of computing resources, and finally r ealize the optimization of user experience Optimization of computing resource utilization and network efficiency.

The management layer of the edge network reports the network information and th e arrangement information of the edge computing power to the arrangement layer of the edge computing power, and accepts the final strategy of the arrangement of the distributed computing power to be implemented. And the edge network managem ent layer reports the edge network status information to the cloud network manageme nt layer, and accepts the distributed network resource arrangement information.

When the edge management platform receives an application request from a user, it will forward the request to the cloud management platform after verifying th e user.
The cloud network management layer reports the network information to the compu ting power scheduling layer, which receives the network information, informs the com puting power service and perceives the computing power status through the computing po wer scheduling layer, so as to generate the computing power route and monitor the r oute in real time. The cloud management platform sends the generated computing power arrangement information to cloud network management. Cloud network management w ill distribute the received computing power arrangement information to edge network management.
Edge network management reports network information and computing power schedul ing information. Edge computing power scheduling performs computing power routing f orwarding operations, and issues the final computing power scheduling strategy through co

mputing
  power routing addressing, computing power routing notification.
Computing resources protect CPU, GPU and other resources to meet the different re
quirements of
 general computing and proprietary computing. These computing resources have grea
ter advantages
 in parallel computing efficiency and low latency computing performance. Heteroge
neous computing s
olutions are deployed based on business requirements to meet the performance requ
irements of computing
, I/O, and network intensive applications.

4. Manageability Considerations

TBD

5. Security Considerations

TBD

6. IANA Considerations

This document requires no IANA actions.

7. References

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

Acknowledgments

TBD

Authors' Addresses

Zhengjie Sun
Beijing University of Posts and Telecommunications

Email: sunzhengjie@bupt.edu.cn

Hui Yang
Beijing University of Posts and Telecommunications

Email: yanghui@bupt.edu.cn

Chao Li
Beijing University of Posts and Telecommunications

Email: lc96@bupt.edu.cn

Sheng Liu
China Mobile

Email: liushengwl@chinamobile.com

Haomian Zheng
Huawei Technologies Co

Email: zhenghaomian@huawei.com