Application Layer Traffic Optimization
Internet-Draft
Intended status: Informational
Expires: January 9, 2017

L. Bertz
Sprint

July 8, 2016

Programmable NFV/SDN orchestration with ALTO
draft-bertz-alto-sdnnfvalto-02

Abstract

This document defines optional Diameter attributes for efficient policy provisioning.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2017.

Copyright Notice

Table of Contents

1. Introduction

This document describes a proposed framework for Orchestration in a Network Function Virtualization (NFV) and Software Defined Networking (SDN) environment. Both technologies have a loose coupling that enables automated deployment of Virtual Network Functions (VNFs).

1.1. Reference Models

Figure 1 shows the ETSI NFV Reference Model.

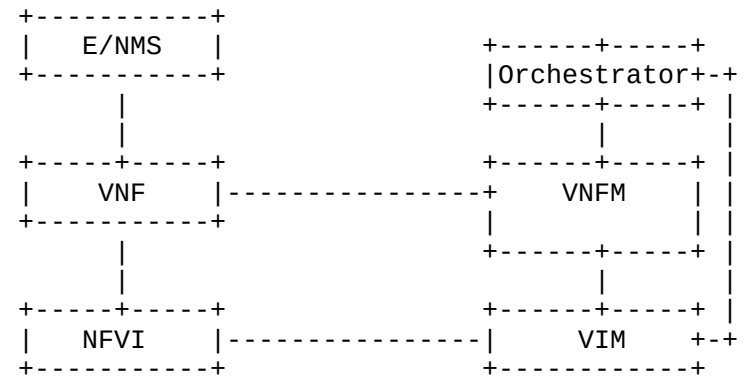


Figure 1: ETSI NFV Reference Model

The Element Manager (EM) and VNF Manager (VNFM) control the VNF. The VNFM interfaces to and controls the VNF for the NFV environment. The Virtual Infrastructure Manager, e.g. OpenStack or OpenVIM controllers, provides control of the underlying hardware infrastructure, i.e. compute, networking, memory and storage. Figure 2 shows the common VIM configuration where an SDN Controller is used to provide the underlying networking infrastructure. Here the SDN Contoller is separated from the VIM and the SDN switch is separated out from the rest of the NVFI.

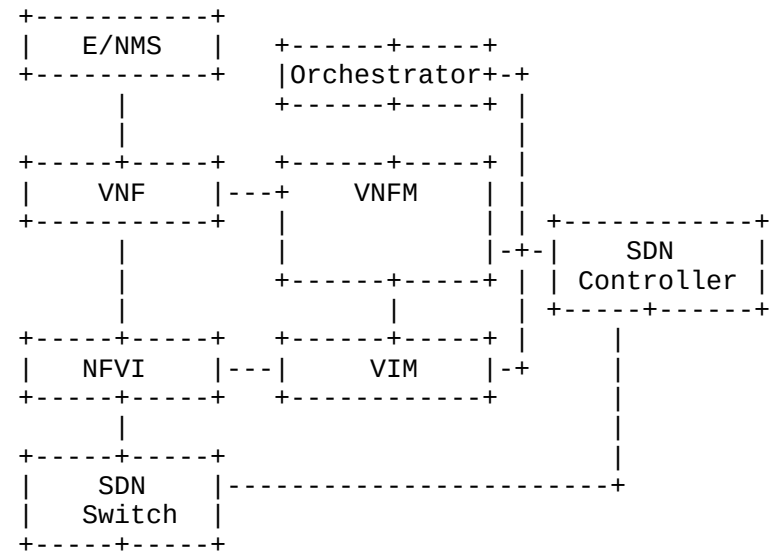


Figure 2: ETSI NFV + SDN Reference Model

The OSM, Open Source MANO (Management And Network Orchestration), group has refined the NFV model in order to address some of the scaling issues. Figure 3 shows a portion of the OSM model.

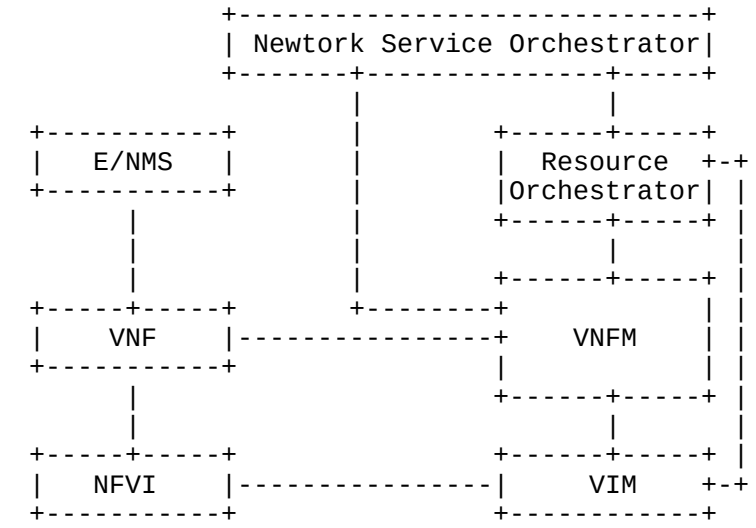


Figure 3: ETSI NFV + SDN Reference Model

The model breaks the NFVO into a Network Service Orchestrator (NSO) and Resource Orchestrator (RO). This separation improves scaling of the NFVO and allows the RO to concentrate on support for multiple VNFMs and VIMs.

1.2. Challenges

Open questions exist with the NFV model around data models, scale and multi-domain support. The industry often points to the success of cloud services with respect to scaling and data models but can’t agree upon how those solutions could be standardized to support NFV. The OSM model improves scaling but does not answer exactly how scalable the solution can be.

Both models assume that data is provided through the VIM to the NFVO components and data required by new VNFs is present in the system. This implies a large amount of information flow in a proactive manner from SDN Controllers through the VIM. If data is required from the Element Manager a direct interface to the VNFM is required but is not specified in either model.

Assumed access to data through the VIM or VNFM is problematic. Relevant data such as latency between sites can be measure via SDN

Controllers supporting Wide Area Networking (WAN). Unfortunately WAN SDN Controllers, due to factors such as separation of concerns, scalability or security, are often not the same SDN Controllers a VIM will communicate with.

The number of SDN Controllers is often underestimated:

SDN Controllers that support low latency signaling must be located close to the SDN switches they provide signaling on behalf of.

SDN Controllers are separated not only by function, e.g. Wide Area, Application Specific or Data Center, but also by Security and Organization.

The number of instances of VIMs and associated SDN Controllers varies but in general VIM Orchestrators manage a small number of racks with many VMs. Such densities produce large information that can limit the scalability of the associated SDN Controllers.

In practice, regional management models, multiple security zones and the fact that dense computing platforms can limit a system such as OpenStack to 4 to 12 racks of computing equipment per instance implies the presence of several SDN Controllers in any one network.

Regardless of a proposed framework for Orchestration, the following challenges exist:

Access to data before a single Orchestration task occurs

Access to data from many SDN and VIM related sources

Access to new kinds of data required for Orchestration of a VNF

Scalability of the data access

2. Requirements Language

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

3. Solution Approach

A good solution that couples SDN and NFV permit Orchestration with a loose coupling of both technologies. To provide a basis for the solution a method is described that approaches how Orchestration may

occur for an Orchestrator looking at a single NFV Forwarding Graph (NFV FG) may determine if a VIM has infrasctructure the meets its requirements.

3.1. Types of Information

Two kinds of information are required for the selection of a set of suitable resources to serve the VNF FG defined for a network service:

Resource Constraints Specific constraints of a resource serving the VNFs of the VNF FG. These constraints include minimum and maximum processing, memory, storage and network requirements. They are based upon available resources, i.e. inventory, available as opposed to a Performance Constraint.

Performance Constraints Specific constraints based upon a measured / computed metric. These measurements may occur over time and can exist outside of any specific VNF activity, e.g. the latency between two sites.

3.2. Single Candidate Solution Determination Process

Looking at the types of Orchestration information, one can evaluate a specific candidate solution for a subgraph in the VNF FG as a set of matrices.

Let VNF Requirements vector, VREQg, be the n Resource Constraints (RC) and m Performance Constraints (PC) for a VNF FG subgraph g

VREQg = [RC1 RC2 ... RCn PC1 PC2 ... PCm]

Let VNF Solution vector, VSOLg, be the vector representing the n Resources (R) and m Performance Metrics (PM) for a VIM’s resources supporting VNF FG subgraph g

VSOLg = [R1 R2 ... Rn PM1 PM2 ... PMm]

Without loss of generality satisfaction of a specific Performance Metric can be determined using subtraction, e.g. VREQg - VSOLg. For example if for latency must be less than 5 then negating the sign for the PM and associated RC. If any value in the resulting vector is less than 0 then, VSOLg does not satisfy VREQg.

Multiplying the resulting matrix by a M+N x 1 matrix of weights yields a value that can be determined as the fitness of the solution. His can be compared with VSOLg values.

3.2.1. Resource Values as Partitions

The representation of Resource Constraints includes the minimum/ maximum inventory required. However, a VIM representing a resource for a graph to a Resource Orchestrator must be careful not to over-summarize the values of the resources it has. For example, let the VNF FG subgraph G be comprised of 3 VNFs:

- VNF A requires 2 CPUs
- VNF B requires 2 CPUs and
- VNF B and VNF A have an anti-affinity, e.g. cannot be in the same hardware.

A VIM stating it has 4 CPUs of resource is insufficient to determine if it can satisfy G. Looking at the partitions (# of ways to add integers to equal the value of the integer) we see the partitions of 4 are:

4

3+1

2+2

2+1+1

1+1+1+1

Of the 5 partitions of 4 only one, 2+2, would satisfy G. Thus, it is important to represent the paritions and not summarize the value in order to determine if a NFVI has sufficient resources for satisfying the VNF FG.

For representing large numbers though this data can be compacted as we propose here. We let the notation for Resource values be

Resource Vector of Type z for RZ(A) = X, [A], sub-partition summary of X, [sub-partitions of X]

where X is a non-negative integer and partition of X for the entity optionally labelled "A". The sub-partitions of X are themselves partitions. The sub-partition summary is the specific sub-partition of X in comma separated values. while the options sub-partition details may also be present.

If the VIM has in Chassis X 3 blades, B1 with 2 CPU, B2 with 1 CPU and B3 with 1 CPU we represent this as

Rcpu (Chassis X) = [4, "Chassis X", [2,1,1], [[2, "B1", [2]], [1, "B2", [1]], [1,"B3",[1]]]]

When evaluating this solution we can quickly look at the first value and determine basic satisfiability. Further inspection is possible as well.

A multi-dimensional resource representation is achieved by turning individual values into arrays. For example if we added memory for all blades of 4 Gigabytes we could represent the Resources as

R[cpu, memory](Chassis X) = [[4, 6], "Chassis X", [[2,2],[2,1],[2,1]] [[[2,2], "B1",[2,2]], [[1,2],"B2", [1,2]], [[[1,2],"B3",[1,2]]]]

Other compaction notations can be introduced as well, e.g. representing B2 and B3 as {2}[2,1] in the partition description and [[1,2], ["B2","B3"], [1,2]]. However, it is proposed that this is the maximum amount of information for Resource values interchanged between VIMs and Orchestration functions.

When no anti-affinities are present it may be sufficient to look at the initial values of the Resource vector. As the example shows looking at specific partition information is required when anti-affinities are present in the VNF FG subgraph.

NOTE that although this document proposes a representation for Resource Values other representations exist that are valid, e.g. a tree repsresentation, and may be more efficient.

3.2.2. Performance [Metric] Constraints and Measurements

Performance Constraints for VNF FG subgraph G have value and optionally measurement constraints. Some measurement constraints are:

Staleness: Age of the measurement that is suitable for usage; otherwise new measurements must be taken.

Granularity: Fine or coarse grain. Fine grained measurements require direct measurements between the endpoints (resources of underlying resources such as the hardware) in question. Coarse grain measurements require values produced through measurement of entities in the same logical entity, e.g. site, chassis, rack, etc., of the proposed solution elements.

Accuracy: The degree of closeness of measurements to its true value.

3.2.3. Data Required

Table 1 shows the data required and source(s) in order to determine VIM solution candidate suitability for a VNF FG subgraph.

Data	Data Structures	Source Element(s)
VNF FG Subgraph Resource Requirements	VNF Descriptors for VNFs in the sub-graph	Resource / Network Service Orchestrators
VNF FG Subgraph Performance Constraints	VNF Descriptors for VNFs in the sub-graph	Resource / Network Service Orchestrators
VIM Partitions (for candidate solutions)	VIM Resource Vectors	VIM
VIM Partition associated Performance Measurements	Metric Information	SDN Controllers, VIMs, VNFs, Element Managers

Table 1: Data, Structures and Sources for VREQg and VSOLg

3.3. Providing Performance Metrics Scalably via ALTO

ALTO is proposed, with some modifications, to provide the Performance Measurements. This includes:

FEATURE 1 Permit the Orchestrating clients (ALTO Clients) to determine if the measurement constraints are satisfied by data provided.

FEATURE 2 Provide a consistent interface from multiple data sources for metrics.

FEATURE 3 Provide mechanisms where measurements available but not previously provided by the ALTO system can be supplied.

FEATURE 4 Provide the ability to initiate measurements not currently made for a metric between two entities.

FEATURE 5 Provide mechanisms where new metrics not currently measured can be provided.

3.3.1. Why ALTO?

The Application Layer Transport Optimization (ALTO) standards provide network map, metric and property information for Endpoints and containing logical entities known by their Provider Identifiers (PIDs). Although PIDs are most often thought of as networks, network partitions or sites, they can be as granular as virtual devices or a blade hosting multiple virtual machines. Endpoints are contained in PIDs. PIDs may be contained by other PIDs.

Cost Maps are based upon metrics and services provide both fine (endpoint to endpoint) and coarse (pid to pid) grain measurements.

Pull (ALTO base RFC) and push (Server Side Events) mechanisms exist for ALTO Clients.

New metrics, maps and measurement data can be supported without change to the Client. However, these mechanisms need further definition to allow Orchestrators, acting as ALTO Clients, to manipulate the ALTO system to meet is needs.

The biggest advantage of ALTO is its ability to serve what a specific application needs for optimization while supporting multiple applications simultaneously through a common, REST based interface. In general, VNFs and Services represents multiple applications and the supporting Orchestrators can use ALTO as a single protocol for the acquisition of performance information.

3.3.2. Feature 1 Support

Given the use of an HTTP RESTful interface, the use of HTTP Date, ETag, Cache-Control Expires and Last-Modified provide information regarding the timing information associated with a request. Use of the Cache-Control, If-Since and If-Unmodified-Since allow Client based control for Staleness.

Given the support of Endpoint (fine grain) Cost Maps and general Cost Maps (coarse) the Client can use the specific ALTO service it desires for granularity.

The accuracy of specific measurements can be provided as another metric, placed as a summary value of a property in the Cost Metric’s associated Information Resource Directory entry or provided as an Endpoint/PID property.

3.3.3. Feature 2 Support

To support multiple sources and Clients (Orchestrators) the following considerations are made.

SDN Controller can provide:

- Network Map
- Cost Map (PID to PID costs)
- Endpoint Properties for those Endpoints it owns or property data about an Endpoint
- Endpoint Costs - For the Endpoints it owns these are Fine-grained costs
- PID resolution for the SDN Controller - We represent this in the map by using a PID that is a 'subdomain' of the network map(s) it is a part of. (This implies some extra metadata for the SDN Controller in the VNF Descriptor or configuration).
- OPEN ISSUE: Multiple Servers providing the same JSON Path values in ALTO (when Controllers are Master/Slaves or primary/backups)?

NFVO can provide

- Network Map (unlikely - it should be configured to perform IRD delegation which punts this function to another ALTO Server)
- Cost Map (unlikely - it should be configured to perform IRD delegation which punts this function to another ALTO Server)
- Endpoint Properties - The data from the VNF Record for IP addresses in it
- Endpoint Costs - Yes, if present for a VNF, not already provided by the SDN Controller and is fine grained only
- OPEN ISSUE: How to detect that SDN Controller and NFVO are providing same metric for same Endpoint(s), especially NOT re-presenting data from SDN Controller sent via the VIM?

Although the NFVO was chosen, 4 possible routes were analyzed regarding how to get VNF data to an ALTO Server using the ETSI model.

1. VNF => VNFM => NFVO

2. VNF => CDN Controller (NO STANDARD)

3. VNF => EM acting as ALTO Server

4. VNF is an ALTO Server

NFV Infrastructure (NFVI) data is delivered to NFVO (NFVI => VIM => NFVO)

The MOST desirable paths is #1 because the NFVO can provide VNF and NFVI data.

However, if high frequency, hysteresis, etc for the VNF is desired then Option 3 (use of EM as an ALTO Server) but the Client may still need access to the NFVI data from VIM or NFV

The NFVO is still the best option

- It must watch performance data of VNF and NFVI to determine if new instances of the VNF should be instantiated
- Both ALTO clients and NFVO are interested in ONLY significant events

To scale, some form of ALTO aggregation will be required.

- Client side integration of many ALTO Servers is complex and defeats the ease of Service ALTO provides
- There will be multiple domains and with filters people are likely to share data
- The number of distinct data sources is quite high - SDN Controllers, VIMs, VNFs, Element Managers

Aggregation of partial ALTO information is possible but not standard. Figure 4 shows the proposed system uses ALTO Client standards and ALTO Server Side Events (SSE) [I-D.roome-alto-incr-update-sse] to maintain synchronized state between ALTO Client and Server.

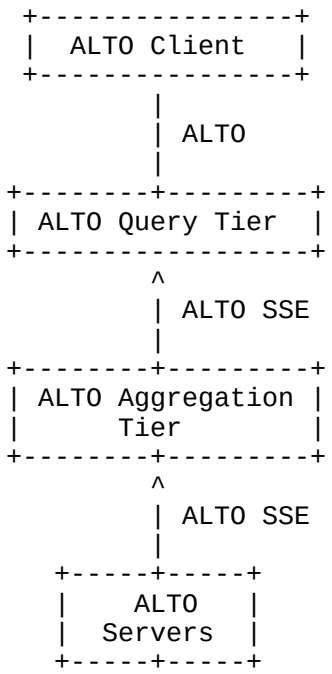


Figure 4: ALTO Aggregation

Figure 5 shows the proposed Aggregation overlay with SDN and NFV.

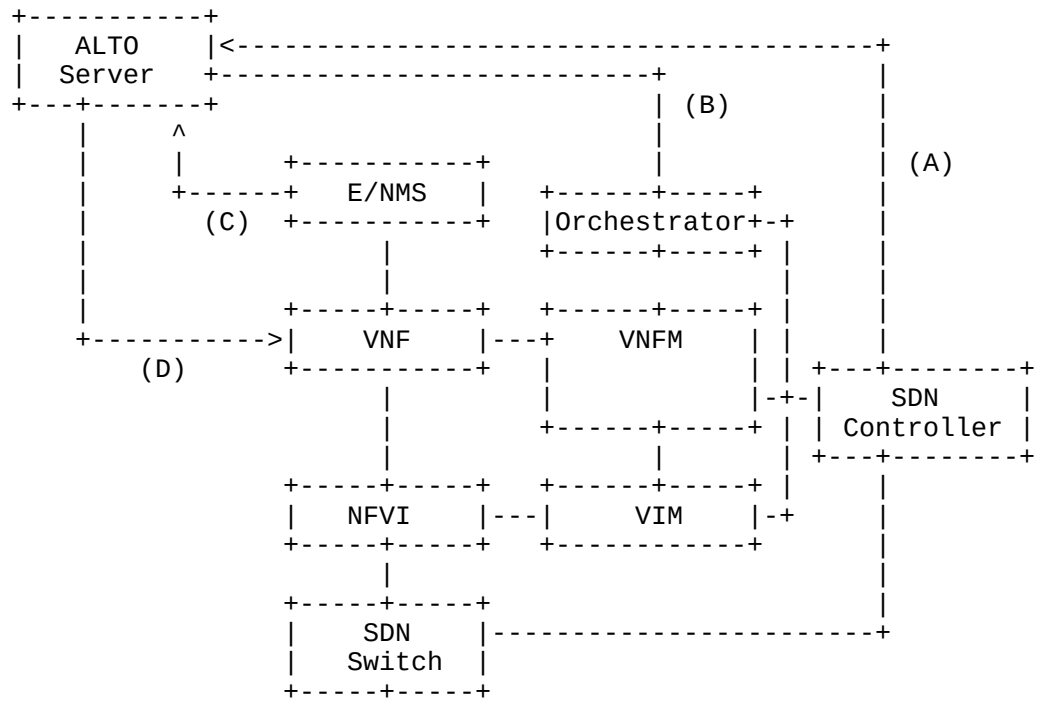


Figure 5: ALTO, SDN and NFV

Links A, B and C use ALTO SSE [I-D.roome-alto-incr-update-sse] to update an ALTO Server. Link D, if present, uses the ALTO protocol [RFC7285].

Links B and C ALTO Servers delegate their Map, Cost and EPCS services to the ALTO aggregators. Delegation is pushed to IRDs of the Query Tier (see Figure 4). This allows other ALTO Clients to query and Element Manager or NFVO function and be directed to the consolidated view provided by the ALTO Aggregator.

Link B also includes an ALTO Client on the Orchestrator that uses the ALTO Servers to aqcuire information for it’s processing.

The SDN Controller used Link A to provide the Map Service, Cost Service, Endpoint Costs and Property Services as required.

3.3.4. Feature 3 Support

Support for Feature 3 was briefly described in [I-D.bertz-alto-aggrimpl] as ALTO Demanded Query Recognition but is expanded upon here as On-Demand measurement. The proposed feature

leverages ALTO's use of HTTP by looking at query misses. Given enough misses, e.g. a threshold, the ALTO Server will take this as a hint that gathering the data is important to the Client.

This opens opportunity to add filtering to the ALTO Server to limit the amount of metric data or provide a generic filter for initial configuration of a Server or Aggregator. The generic filter could

- Be returned during a query
- Pushed via Server Side Events (SSE)
- Allows for fast restart of a server since it is like a cache

Network Maps are NOT considered filterable at this time.

Adjustable filters, not discussed initially [I-D.bertz-alto-aggrimpl], could be used in the solutions to only send Cost information that Clients ask for. Initial filters could be empty - only sending network topology. The network image of ALTO Servers and Aggregator VNFs, i.e. ALTO related configurations in a stock VM / KVM / sbare metal image, can be set with a common 'blank' filter; reducing configuration complexity.

High Client Query rates could further be capitalized upon by the Server to

- Take more measurements from different endpoints to increase samples / improve computations
- Increasing frequency of measurements

Aging of old queries could result in narrowing filters.

3.3.5. Feature 4 Support

Adjustable Filters do not address how measurements could be initiated between two entities to generate cost map data (assumes data is already present in underlying data source.

A new ALTO feature, Measurement Initiation, is proposed. It is ability to initiate measurement process (by entities outside of the ALTO Server). It solves the 'what if data is not present' question but opens new questions

- How to detect Measurement Initiators?
- How to let Measurement Initiators find each other?

- Where to do work (outside of the current ALTO scope)?

- What is the protocol to initiate measurement?

The proposed feature measures data currently being asked for that does not exist in the local ALTO server. It determines who should measure by relying on Endpoint and especially PID properties. A 'Measurement Initiators' property is added to Endpoints or, preferably PIDs, which is a list specifying the endpoints that could initiate measurements to/from a PID or endpoint. Additional discriminators such as the metrics supported by specific initiators can be provided. Ideally, the endpoints for a measurement initiator can be added to the Information Resource Directory (IRD) metric information.

The process is defined as:

1. Subsequent ALTO Server query misses occur even though an Adjustable Filter exists.

2. The ALTO Server sends request to Measurement Initiator with the Endpoints / PIDs and metric(s) requested. This instance is referred to as the Initial Measurement Initiator (IMI).

3. IMI then queries ALTO Endpoint Property Service to find Measurement Initiators for corresponding Endpoint Property / PIDs. These are Corresponding Measurement Initiators (CMIs).

4. IMI and CMIs arrange measurements and populate data into local data stores accessible by their local ALTO Server(s)

An ALTO Server (MI Client) can also ask for

- Measurement Frequency increase / decrease

- Cessation of a measurement

- More sampling points an aggregate computation, e.g. PID level metrics

The specific Measurement Initiation, Negotiation and subsequent lifecycle management is outside of the current ALTO scope and requires further work.

3.3.6. Feature 5 Support

In order to support new metrics a plug-in model to ALTO servers can be supported. When a VNF Descriptor is introduced into the system, ideally, plug-ins supporting metric measurement, computation and the Measurement Initiation process (Feature 4) will be made available.

When the ALTO server determines that it is time to provide the new metric, it can download the appropriate plug-ins, update its IRD and begin processing. There may be numerous triggers for this.

Being Proactive

There are several situations where the system can be proactive.

- When a VNF Descriptor is introduced into the system an Orchestrator can proactively query for measurements to 'prime' the ALTO system.
- When VIMs appear to be close to exhaust the ALTO Client layer can search for suitable VIMs to focus their provisioning upon and begin filter adjustments and demand measurement processes.
- Upon the addition of a new VIM, SDN Controller or other ALTO source the ALTO Client layer can execute priming queries to begin the Measurement Initiation and On-Demand Measurement processes.

3.3.7. Summary

The use of ALTO, with some modifications, allows performance measurements to be dynamically added to the system for new sites, endpoints or even new types of performance metrics. Enhancement to ALTO are required but appear to be minor.

4.

ALTO provides the ability for Clients to look up ALTO Servers [RFC7285]. This can be enhanced to support Aggregator Discovery or an alternative method can be constructed. ALTO information origins, which are ALTO servers, can be discovered by Aggregators acting as Clients. The mechanism by which Aggregators divide work can be defined, if required. Regardless, as new Aggregators and Servers are instantiated they can be found and self-organize.

5. Programmability Overview

Using the Single Candidate Solution Process described above multiple subgraphs can be evaluation across multiple VIMs, i.e. many VREQg / VSOLg pairs can be computed for different subgraphs using matrix operations. By partitioning the VNF FG into subgraphs a total VNF FG can be evaluated across multiple VIMs. Figure 6 shows the overall process.

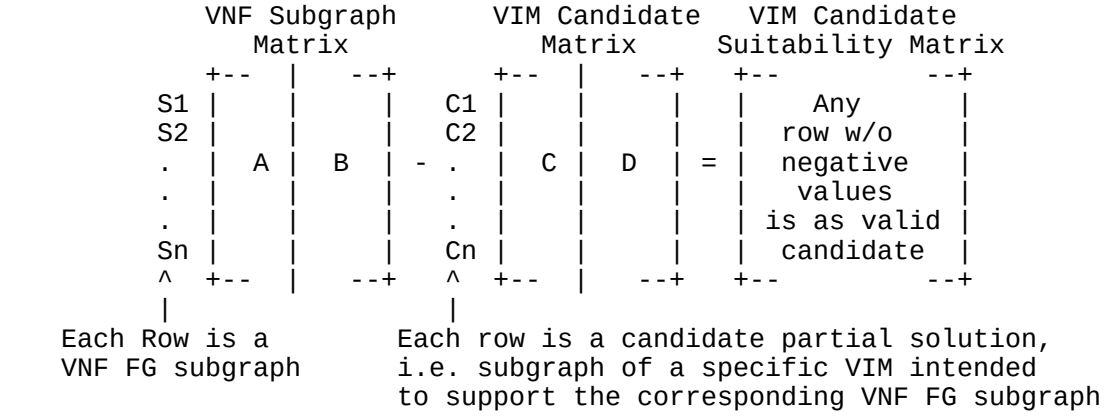


Figure 6: Parallel VNF FG subgraph and VIM candidate Evaluation

Each row of the subgraph matrix is a VREQsi for the specified subgraph Si. Each row of the VIM Candidate Matrix is VSOLci, a corresponding solution subgraph in a VIM's subgraph Ci. RC data and Performance data are provided as described above. Like the single candidate solution process, any resulting row with a negative value is in invalid solution.

Figure 7 shows the weighting process, assuming the VIM Candidate Suitability Matrix results in m valid candidates.

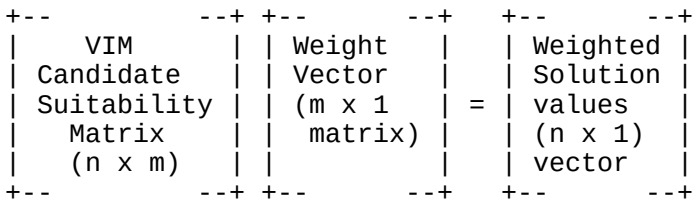


Figure 7: VIM Candidate Suitability Matrix Weighting

The Orchestrator must then reconstruct candidates that create a proper super digraph of the VNF FG. The union of some solutions *may* result in graphs equal to the VNF FG while others will be super digraphs with distinct advantages.

The overall process for the Orchestration layer is defined as:

STEP 1 Orchestrator generates partial solutions, i.e. subgraph, of the VNF FG that it believes it can be satisfied by the VIMs' resource information. These become candidate partial solutions Si for a subgraph of the VNF FG.

STEP 2 Orchestrator computes the Performance Constraints for the VNFs contained in the subgraph as well as those introduced by the partitioning of the VNF FG into the specified subgraph Si.

The union of data in Step 1 and 2 completes the initial Subgraph Matrix.

STEP 3 Orchestrator determines the possible VIM Candidate Solutions Ci for a specific Si. Note that here multiple VIMs or solution within a VIM may be evaluated for a specific Si. In this case the Subgraph Matrix is expanded accordingly and results in multiple copies of Si appearing in the matrix. How or even if this is done is specific to the Orchestrator. The Resource Partition(s) are then placed into each row.

STEP 4 Orchestrator queries ALTO aggregators for Performance Measurements associated with the VIM Candidate Solution in each row.

The addition of rows, as required, in the Subgraph Matrix in Step 3 completes its construction.

The union of data in Step 3 and 4 completes the VIM Candidate Matrix.

STEP 5 The VIM Candidate Matrix is subtracted from the Subgraph Matrix creating the VIM Candidate Suitability Matrix.

STEP 6 Any row with a negative value is removed from the VIM Candidate Suitability Matrix.

STEP 7 (Optional) A weight vector is multiplied to the VIM Candidate Suitability Matrix resulting in an objective function evaluation of each VIM Candidate.

STEP 8 The results of Step 7 (or Step 6 if no weighting was applied) is then recombined with various subgraphs that could provide a super digraph of the VNF FG. This value is then evaluated and a final set of VIM candidates is made.

STEP 9 All VIMs (or their Resource Orchestrators) are sent requests for fulfillment of the VNF FG.

6. Data Provided in Orchestration Requests

In order to provide more programmability the following data is sent from the requestor:

Request Row VNF FG Subgraph’s Requirements.

VIM Data Row It shows the ALTO (performance) + Resource Information that was used to determine the selection.

Upper Bounds Row Upper Bounds that the VIM may auto-scale the VNFs in the VNF FG subgraph to w/o querying the VIM.

Lower Bounds Row Lower Bounds where if they are maintained for a pre-defined period will require the VIM to notify the Orchestrator.

The VIM data row gives the request receiver insight into the accuracy/staleness of Resource Information and/or Performance Measurements. If these values are considered too old it is up to the receiver of the request to increase the update frequency. If information is consistently wrong over multiple requests it may also be a bad actor in the system (defective or a security issue) that should be raised by the request receiver.

The upper and lower bounds provide thresholds for meaningful events to the requestor. If the receiver does not monitor all of the performance metrics in a bound row it may reconfigure itself to do so or merely concern itself with what is monitors. Negotiation of what it will monito is outside of the scope of this document.

7. Proposed System and Benefits

The system is comprised of the following elements:

- ALTO Server integrated SDN Controller
- ALTO aggregators
- Measurement Initiators (may be part of Controllers)
- ALTO integrated NFVO
- ALTO integrated VNF EM (optional)
- ALTO Client based Orchestrator

The system contains the following features:

- ALTO aggregation for Scaling over multiple SDN Controllers / NFVI domains
- On Demand Measurements / Measurement Initiation to adapt to different Orchestrator information needs
- Minimal interchange between VIM and Orchestrator
- Orchestrator can get data from ANY ALTO source
- System can
  - Adapt to new metrics from VNFs
  - Send only data that is required
  - Auto discovery of VIMs, Aggregators and Orchestrators via ALTO self organization
- Detection of stale or incorrect information

Multiple VIMs, Orchestrators and VNFs can be dynamically supported.

- Orchestrators can cut the VNF FG over multiple VIMs, create VIM candidate solutions and computations w/o issue
  - How the VNF FG is cut is Orchestrator specific
  - How candidates are computed given VIM data is Orchestrator specific

    * Orchestrators can meaningfully differentiate

  o VIMs can serve multiple Orchestrators consistently

  o VNFs that introduce new performance metrics can be accommodated

    * Use On Demand Measurements for specifying metric information w/ o system overload

    * Use Measurement Initiation to provide metric information that was not currently being captured

    * Provide some 'plug-in' framework in popular SDN Controllers to dynamically load / compute metrics, e.g. javascript scripts with Controller API access

    * When an Orchestrator gets a metric is has not seen before it could proactively initiate the On Demand Measurement / Measurement Initiation functions to ensure system can make an informed decision the first time a VNF instantiation request is made

8. Summary

A proposed framework for Performance Measurement and Resource information exchange amongst various data sources can be used for determination of VNF FG instantiation. This framework can adapt to new measurements between entities and even the introduction of new metrics that must be measured. Self-organization and the ability to expand / contract the information interchanged make the solutions data minimal and relevant. Innovation in orchestration is still possible but the high level types of information, what is interchanged and how that is exchanged are standardized.

9. IANA Considerations

This memo includes no request to IANA.

10. Security Considerations

This informational document relies upon security mechanisms, if any, that would recommended by ALTO specifications.

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, <http://www.rfc-editor.org/info/rfc2119>.

[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, <http://www.rfc-editor.org/info/rfc7285>.

11.2. Informative References

[I-D.bertz-alto-aggrimpl] Bertz, L., "Extensions for ALTO Aggregation", draft-bertz-alto-aggrimpl-00 (work in progress), October 2015.

[I-D.roome-alto-incr-update-sse] Roome, W., Shi, X., and Y. Yang, "ALTO Incremental Updates Using Server-Sent Events (SSE)", draft-roome-alto-incr-update-sse-02 (work in progress), March 2015.

Author’s Address

Lyle Bertz
Sprint
6220 Sprint Parkway
Overland Park, KS 66251
United States

Email: lylebe551144@gmail.com

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

K. Gao
Tsinghua University
J. Zhang
Tongji University
Y. Yang
Yale University

July 8, 2016

ALTO Flow Cost Service
draft-gao-alto-fcs-00.txt

Abstract

OpenFlow [openflow] is the current standard southbound protocol for Software-Defined Networking. In this document, we define a new service, namely the Flow Cost Service, for clients in a OpenFlow-enabled network to query the network information.

Requirements Language

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

Status of This Memo

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

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

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

This Internet-Draft will expire on January 9, 2017.

Copyright Notice

Table of Contents

1. Introduction

With the emerging technologies in the data plane, where multiple header fields can be used to determine the forwarding path, networks are moving to more flexible routing mechanism beyond the simple destination-based routing. As a consequence, the endpoint cost service (ECS), which depends on only source and destination IP addresses as currently defined, is no longer sufficient to provide accurate cost information.

In this document, we consider the extension of ALTO service which provides the cost service, for networks using flow-based routing such as Software-Defined Networks using OpenFlow switches. The flow-based routing, in general, provides a more fine-grained control over the packets than destination-based routing. Consider those packets from a specific source to a specific destination. With destination-based routing, these packets will always use the same path, and hence ECS can provide the same routing cost. Flow-based routing, however, can partition these packets into multiple subsets (flows), where different subsets (flows) can go through different paths, and hence have different routing cost values. For example, large science data flows may go through a DMZ path with low cost, and other traffic may need to go through more security checks, with higher costs. Although one may still use ECS, which may provide an aggregated cost (e.g., average), the result can be inaccurate and misleading.

To satisfy the growing demand of obtaining accurate costs in a network using flow-based routing, a new ALTO service named the flow cost service (FCS) is defined.

2. Basic Data Types

The flow cost service introduces some new basic data types, as defined below.

2.1. Flow ID

A flow ID has the same format as a PIDName, as defined in [RFC7285] Section 10.1 [1]. It is used to uniquely identify a flow in a flow cost service request.

2.2. Typed Header Field

A typed header field represents a particular field in a network protocol that can be obtained at the application layer. It is represented by the protocol name and the field name, concatenated by the colon (':', U+003A). The typed header fields are case insensitive.

For example, "ipv4:source" and "IPv4:source" both represent the source address field used in IPv4 and "tcp:destination" represents the destination port for a TCP connection.

See Table 2 for a list of proposed typed header fields.

2.3. Cost Confidence

A cost confidence is defined as a JSON integer within the range of [0, 100]. It represents the ALTO servers' estimation on the accuracy of the returned costs. The larger the cost confidence is, the more accurate the path cost SHOULD be. If the cost value is very accurate, for example, a unique path can be identified for a flow with the provided information, the ALTO server SHOULD provide a cost confidence of 100.

The cost confidence CAN be used as an evidence of ambiguous paths, which is often associated with insufficient information in a query. If the ALTO clients find the associated cost confidence value is low, it can narrow down the flow header space in the query by adding optional fields or use IP addresses instead of prefixes.

The cost confidence value can be computed in several ways. For example, ALTO servers MAY use the following formula for some cost metrics:

```
    c = 100 * (1 - |deviation / mean|)

                  0        if c <= 0
    confidence =
                  round(c) if c > 0
```

where mean and deviation are computed from the cost values of all possible paths.

3. Flow Cost Service

A flow cost service provides information about costs for each individual flows specified in the requests.

3.1. Media Type

The media type of the flow cost service is "application/alto-flowcost+json".

3.2. HTTP Method

The flow cost service is requested using the HTTP POST method.

3.3. Accept Input Parameters

The input parameters of the flow cost service MUST be encoded as a JSON object of type FlowCostRequest in the body of an HTTP POST

request. The media type of the request MUST be "application/alto-flowcostparams+json".

```
object {
  FlowFilterMap     flows;
} FlowCostRequest : MultiCostRequestBase;

object {
  [CostType         cost-type;]
  [CostType         multi-cost-types<1..*>;]
  [CostType         testable-cost-types<1..*>;]
  [JSONString       constraints<0..*>;]
  [JSONString       or-constraints<0..*><0..*>;]
} MultiCostRequestBase;

object-map {
  FlowId -> FlowFilter;
} FlowFilterMap;

object-map {
  TypedHeaderField -> JSONValue;
} FlowFilter;
```

flows: A map of flow filters for which path costs are to be returned. Each flow filter is identified by a unique FlowId, as defined in Section 2.1. The value types of a field is protocol-specific, see Table 3 for the value types associated with typed header fields in Table 2.

cost-type: The same as defined in [I-D.ietf-alto-multi-cost] Section 4.2.2 [2].

multi-cost-types: The same as defined in [I-D.ietf-alto-multi-cost] Section 4.2.2 [3].

testable-cost-types, constraints, or-constraints: The same as defined in [I-D.ietf-alto-multi-cost] Section 4.2.2 [4].

3.4. Capabilities

The capabilities of the flow cost service is a JSON object of type FlowCostCapabilities:

```
object {
  TypedHeaderField  required<1..*>;
  [TypedHeaderField optional<1..*>;]
} FlowCostCapabilities :  FilteredCostMapCapabilities;
```

with fields:

required: A list of required typed header fields. These fields are essential to find the path cost for a given flow and MUST be provided in a flow filter.

optional: A list of optional typed header fields. The ALTO server MAY leverage the values of the optional fields to find more accurate costs.

3.5. Response

The "meta" field of a flow cost response MUST contain the same cost type information as defined in [I-D.ietf-alto-multi-cost] Section 4.2.3 [5].

The data component of a flow cost service is named "flow-cost-map", which is a JSON object of type FlowCostMap:

```
object {
  FlowCostMap     flow-cost-map;
  [FlowCostMap    flow-cost-confidences;]
} FlowCostResponse : ResponseEntityBase;

object-map {
  FlowId -> JSONValue;
} FlowCostMap;
```

flow-cost-map: A dictionary map with each key (flow ID) representing a flow specified in the request. For each flow, the cost MUST follow the format defined in [I-D.ietf-alto-multi-cost] Section 4.2.3 [6].

flow-cost-confidences: A dictionary map with each key (flow ID) representing a flow specified in the request. For a single cost, the cost confidence for each flow MUST follow the specification in Section 2.3. If the query is using multiple costs where the costs are returned as a JSONArray, the cost confidence MUST also be a JSONArray where each element represents the cost confidence value computed for the corresponding cost type.

3.5.1. Ambiguous Paths

Since new forwarding abstractions support fine-grained routing, for example, OpenFlow 1.5 [OF15] has defined 38 header match fields, it is possible that the ALTO server cannot determine the path using the provided header fields. The computation for costs with ambiguous paths is implementation-specific, the servers can choose to return an

integrated result of all possible paths, or simply use the cost of a random path. The ALTO servers SHOULD provide cost confidences to justify the accuracy of the provided cost values.

The ALTO server SHOULD be able to determine a unique path when all the optional typed header fields are provided without masks for a flow, however, the client SHOULD NOT assume this always holds.

3.6. Errors

The ALTO servers can provide more information to the clients when requests have errors. The FlowCostErrorMap below can provide basic information about two most common errors for the flow cost service. The ALTO servers MAY include it as the data component of an ALTO error response. If multiple errors are identified, the ALTO server MUST return exactly one error code according to [RFC7285] Section 8.5.2 [7].

```
object-map {
  FlowId -> FlowCostError;
} FlowCostErrorMap;

object {
  [TypedHeaderField conflicts<2..*>;]
  [TypedHeadreField missing<2..*>;]
  [TypedHeaderField unsupported<1..*>;]
} FlowFilterError;
```

conflicts: A list of conflicting typed header fields. See Section 3.6.1 for details.

missing: A list of missing typed header fields. See Section 3.6.2 for details.

unsupported: A list of unsupported typed header fields. See Section 3.6.3 for details.

3.6.1. Conflicts

Some header fields may have conflicts. For example, IPv4 fields and IPv6 fields can never appear in the same packet, nor can TCP and UDP ports. These header fields MUST not be included in the same flow filter, otherwise the ALTO server MUST return an ALTO error response, with the error code "E_INVALID_FIELD_VALUE". As specified in [RFC7285] Section 8.5.2 [8], the ALTO server MAY include the "field" and the "value" in the "meta" field. In this case, the ALTO server MUST use the flow ID as the "field" and the flow filter as the "value". However, the recommended approach is to use the

FlowCostErrorMap, where the server CAN provide the conflicting typed header fields in the "conflicts" field of the FlowFilterError associated with the corresponding flow ID.

3.6.2. Missing Fields

The "E_MISSING_FIELD" error code is originally designed to report the absence of required JSON fields. In the flow cost service, the required typed header fields are implementation-specific and the ALTO servers MUST declare the required fields in the capabilities. If any required header field is missing, the ALTO server MUST return an ALTO error response, with the error code "E_MISSING_FIELD". The ALTO server CAN follow the steps defined in [RFC7285] Section 8.5.2 [9] to indicate the location of the missing field. An alternative approach which is also recommended, is that the server provide the missing typed header fields in the "missing" field of the FlowFilterError associated with the corresponding flow ID.

3.6.3. Unsupported Fields

If a query contains unsupported typed header fields, e.g., those not in the "required" nor the "optional" capabilities, the ALTO server MUST return an ALTO error response, with the error code "E_INVALID_FIELD_VALUE". Like how the conflicting header fields are handled in Section 3.6.1, the ALTO servers CAN report unsupported typed header fields in the "unsupported" field associated with the corresponding flow ID.

3.7. Example

```
POST /flowcost/lookup HTTP/1.1
HOST: alto.example.com
Content-Length: 521
Content-Type: application/alto-flowcostparams+json
Accept: application/alto-flowcost+json,application/alto-error+json

{
  "cost-type": {
    "cost-mode": "numerical",
    "cost-metric": "routingcost"
  },
  "flows": {
    "l3-flow": {
      "ipv4:source": "192.168.1.1",
      "ipv4:destination": "192.168.1.2"
    },
    "optional-l3-flow": {
      "ipv4:source": "192.168.1.1",
      "ipv4:destination": "192.168.1.2"
      "ethernet:source": "12:34:56:78:00:01",
      "ethernet:destination": "12:34:56:78:00:02"
    },
    "l3-flow-aggr": {
      "ipv4:source": "192.168.1.0/24",
      "ipv4:destination": "192.168.2.0/24"
    }
  }
}
```

```
HTTP/1.1 200 OK
Content-Length: 312
Content-Type: application/alto-flowcost+json

{
  "meta": {
    "cost-type": {
      "cost-mode": "numerical",
      "cost-metric": "routingcost"
    },
  },
  "flow-cost-map": {
    "l3-flow": 10,
    "l3-flow-aggr": 50
    "optional-l3-flow": 5,
  },
  "flow-cost-confidences": {
    "l3-flow": 70,
    "l3-flow-aggr": 40,
    "optional-l3-flow": 90
  }
}
```

4. Security Considerations

This document has not conducted its security analysis.

5. IANA Considerations

This document defines two new entries to be registered to application/alto-* media types.

5.1. Media Types

This document registers two media types, listed in Table 1.

Type	Subtype	Specification
application	alto-flowcost+json	Section 3.5
application	alto-flowcostparam+json	Section 3.3

Table 1: ALTO FCS Media Types

Type name: application

Subtype name: This document registers two subtypes, as listed in Table 1.

Required parameters: n/a

Optional parameters: n/a

Encoding considerations: Encoding considerations are identical to those specified for the "applicatoin/json" media type. See [RFC7159].

Security considerations: Security considerations are identical to those specified in [RFC7285] Section 15 [10].

Interoperability considerations: n/a

Published specification: This document is the specification for these media types. See Table 1 for the section documenting each media type.

Applications that use this media type: ALTO servers and ALTO clients with the extension to support the flow cost service, either standalone or embedded within other applications.

Additional information: n/a

Person & email address to contact for further information: See Authors' Addresses.

Intended usage: COMMON

Restrictions on usage: n/a

Author: See Authors' Addresses.

5.2. Header Field

TBD: Create the "ALTO Header Field Name Registry".

6. Acknowledgement

7. References

7.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

7.2. Informative References

[I-D.ietf-alto-multi-cost] Randriamasy, S., Roome, W., and N. Schwan, "Multi-Cost ALTO", draft-ietf-alto-multi-cost-02 (work in progress), June 2016.

[I-D.wang-alto-ecs-flow] Shen, X., Zhang, J., Wang, J., and Q. Xiang, "ALTO Extension: Endpoint Cost Service for Flows", draft-wang-alto-ecs-flows-01 (work in progress), April 2016.

[OF15] Foundation, O., "Openflow switch specification v1. 5.0", 2014, <https://www.opennetworking.org/images/stories/downloads/sdn-resources/onf-specifications/openflow/openflow-switch-v1.5.0.noipr.pdf>.

[openflow] McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., and J. Turner, "Openflow: enabling innovation in campus networks", 2008.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[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, <http://www.rfc-editor.org/info/rfc7285>.

7.3. URIs

[1] https://tools.ietf.org/html/rfc7285#section-10.1

[2] https://tools.ietf.org/html/draft-ietf-alto-multi-cost-02#4.2.2

[3] https://tools.ietf.org/html/draft-ietf-alto-multi-cost-02#4.2.2

[4] https://tools.ietf.org/html/draft-ietf-alto-multi-cost-02#4.2.2

[5] https://tools.ietf.org/html/draft-ietf-alto-multi-cost-02#4.2.3

[6] https://tools.ietf.org/html/draft-ietf-alto-multi-cost-02#4.2.3

[7] https://tools.ietf.org/html/rfc7285#section-8.5.2

[8] https://tools.ietf.org/html/rfc7285#section-8.5.2

[9] https://tools.ietf.org/html/rfc7285#section-8.5.2

[10] ttps://tools.ietf.org/html/rfc7285#section-15

Appendix A. Tables

Protocol	Field Name	Description
Ethernet	source	The source MAC address
	destination	The destination MAC address
	vlan-id	VLAN-ID from 802.1Q header
IPv4	source	IPv4 source address
	destination	IPv4 destination address
IPv6	source	IPv6 source address
	destination	IPv6 destination address
TCP	source	TCP source port
	destination	TCP destination port
UDP	source	UDP source port
	destination	UDP destination port

Table 2: Protocols and Field Names.

Typed Header Field	Acceptable Value Type
ethernet:source	JSONString as MAC address
ethernet:destination	
ethernet:vlan-id	JSONNumber in the range of [1, 4094]
ipv4:source	JSONString as IPv4 address or IPv4 prefix
ipv4:destination	
ipv6:source	JSONString as IPv6 address or IPv6 prefix
ipv6:destination	
tcp:source	JSONNumber in the range of [0, 65535]
tcp:destination	0 serves as a wildcard value
udp:source	
udp:destination	

Table 3: Value Types for Typed Header Fields

Authors' Addresses

Kai Gao
Tsinghua University
30 Shuangqinglu Street
Beijing 100084
China

Email: gaok12@mails.tsinghua.edu.cn

Jingxuan Jensen Zhang
Tongji University
4800 CaoAn Road
Shanghai 201804
China

Email: jingxuan.n.zhang@gmail.com

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

Email: yry@cs.yale.edu

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

K. Gao
Tsinghua University
X. Wang
C. Gu
Tongji University
Y. Yang
Yale University
G. Chen
Huawei

July 8, 2016

ALTO Extension: A Routing State Abstraction Service Using Declarative Equivalence

draft-gao-alto-routing-state-abstraction-03.txt

Abstract

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

Requirements Language

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

Status of This Memo

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

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

Copyright Notice

Table of Contents

1. Introduction

The key services of the ALTO protocol [RFC7285] can be considered as information query services about the routing state of a network. Specifically, a cost map of an ALTO metric allows a network application to look up the end-to-end value of the given metric, for

the routing path(s) from a given source to a given destination. The endpoint cost service provides a similar service.

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

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

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

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

endpoint cost service, leaving the aggregation aspects of using network aggregation maps as future work.

This document is organized as follows. Section 2 replicates the multi-flow scheduling example from [I-D.yang-alto-path-vector]. Section 3 gives an overview of the service, and Section 3.2 gives more details on specifying state equivalence. Section 4 gives the specification of RSADE service. Section 5 and Section 6 discuss security and IANA considerations.

2. The Multi-flow Scheduling Use Case

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

We use an example shown in Figure 1. The network has 7 switches (sw1 to sw7) forming a dumb-bell topology. Switches sw1/sw3 provide access on one side, s2/s4 provide access on the other side, and sw5-sw7 form the backbone. End hosts eh1 to eh4 are connected to access switches sw1 to sw4 respectively. Assume that the bandwidth of each link is 100 Mbps. Assume that the network is abstracted with 4 PIDs, with each representing the hosts at one access switch.

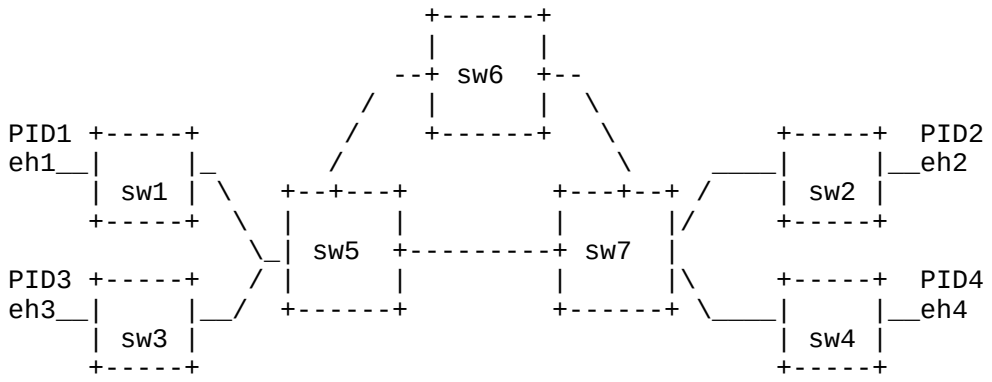


Figure 1: Raw Network Topology

Link	Description
link1	sw1 <==> sw5
link2	sw2 <==> sw7
link3	sw3 <==> sw5
link4	sw4 <==> sw7
link5	sw5 <==> sw6
link6	sw6 <==> sw7
link7	sw7 <==> sw5

Table 1: Description of the Links

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

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

- Case 1: If the two flows use different paths in the current routing state, for example, when the first uses sw1 -> sw5 -> sw7 -> sw2, and the second uses sw3 -> sw5 -> sw6 -> sw7 -> sw4. Then the application will obtain 200 Mbps.

- Case 2: If the two flows share a bottleneck in the current routing state, for example, when both use the direct link sw5 -> sw7, then the application will obtain only 100 Mbps.

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

Now consider what route abstraction can achieve. Assuming case 2 (shared bottleneck), it is sufficient for the network to return a

single abstract link for each flow: ane1(100Mbps), where ane stands for abstract network element, and the number in the number 100Mbps denotes its capacity.

Consider a variation of the preceding case. Assume that the capacity of the link from sw1 to sw5 is 70 Mbps, while the rest are still at 100 Mbps. Then the abstract route from eh1 to eh2 becomes ane1(100Mbps) and ane2(70Mbps).

3. The RSADE Service

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

Another extreme API is that each routing state query provides only the most basic information (i.e., the source and the destination). This, however, does not provide enough information for the routing-state service to compute efficient route abstraction/compression. Hence, the returned routes will be independent of individual functions, missing out opportunities on abstraction or compression.

The routing state abstraction service tries to strike a balance between the two extremes. As a start, such an abstraction service, namely RSADE, is introduced in this document. Using a general abstraction mechanism called the _link elimination_, as described in Section 3.2.1, the service, however, is specialized for multiple-flow coordination, which is the common mathematical model for scenarios from simply using bandwidth as the ALTO routing cost to conducting traffic engineering in a real network.

RSADE is short for Routing State Abstraction using Declarative Equivalence. As the name suggests, this service provides _equivalent_ routing state according to the consumer’s demands which are described in a _declarative_ manner. Figure 2 gives the grammar to specify the query information that a network application sends to the network:

```
rs-query    := flow-desc  equiv-cond

flow-desc   := EndpointFilter  |  FlowFilter
```

Figure 2: Grammar for RSADE

The first component of a RSADE query is the description of the desired flow candidates, which can be either a EndpointFilter as defined in Section 11.3.2.3 from [RFC7285] or a FlowFilter as in Figure 3. A EndpointFilter represents a complete bipartite graph, with _srcs_ on one side and _dsts_ on the other, in a quite compressed way. Meanwhile, FlowFilter enables the application to designate the flows more precisely, which can reduce the computation overhead when the virtual bipartite graph is sparse.

The second component of the query input is the equivalence condition. A particular type of equivalence condition, in the context of multiple-flow coordination, is the range equivalence condition. We give the detailed specification of the condition in Section 3.2.

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

3.1. FlowFilter

Figure 3 provides the grammar of FlowFilter, which is a list of flow specifications each representing one flow in the network. For backward compatibility, a flow specification is described using a (src, dst) pair at the moment. However, it can be extended to support other properties as well, such as VLAN, HTTP proxy server or VPN gateways. The extensions of FlowFilter are beyond the scope of this document.

```
FlowFilter  := flow-list

flow-list   := flow-spec, [flow-list]

flow-spec   := generic-match-condition
```

Figure 3: Grammar for FlowFilter

3.2. The Range Equivalence Condition

3.2.1. Link Elimination

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

Link	R[0]	R[1]	A[1]/bandwidth	A[2]/delay	A[3]
link1	1	0	100M	2ms	...
link2	1	0	100M	5ms	...
link3	0	1	100M	5ms	
Link4	0	1	100M	5ms	
link5	1	1	100M	7ms	
link6	1	1	100M	4ms	

Table 2: Complete Routing State

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

3.2.2. Input

In the general case, the result from the routing-state query will become the input parameters for the algorithms in the network application to help make decisions. Let x denote the vector of the decision variables in the application. Then, one can identify that a generic structure of the application is to solve/optimize an objective function, obj(x), subject to two types of constraints on x: (1) those do not involve the results from the routing state query; and (2) those do. Let the first type limit x in X0. Regarding the second type, most algorithms typically handle only linear constraints, and hence the set S of constraints of this type will be of the format a_k x <= b_k, where a_k is a vector, and b_k a constant. Hence, it is in a_k or b_k where the result from the

routing-state query appears. Let A x <= b as a matrix format to represent the whole set of constraints.

[Equivalent Routing-State Query] A declarative equivalence based routing-state query is one where the querier (application) declares X_0 and a set of constraints S = {a_k x <= b_k}.

Based on the definition of equivalent routing-state query, the grammar for the equivalence condition is defined in Figure 4.

```
 equiv-cond            := variable-list  X0  link-constraint-list

 variable-list         := variable-name[, variable-list]

 X0                    := simple-constraint[, simple-constraint]
 simple-constraint     := simple-expr  CMP-OP  simple-expr
 simple-expr           := constant * variable-name[ + simple-expr]

 link-constraints-list := link-constraint[, link-constraint-list]
 link-constraint       := link-expr  CMP-OP  link-expr
 link-expr             := constant | attribute-name | variable-name
                       |  constant * link-expr
                       |  attribute-name * link-expr
                       |  link-expr + link-expr
```

Figure 4: Grammar for Equivalence Condition

3.2.3. Response

We use the terms defined below to better describe the abstracted routing state.

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

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

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

It can be seen that if the attribute of a link does not appear in a minimal constraint set, this link will have no effect on the querier application’s decision making. In that case, eliminating the link would cause no information loss. Thus to minimize the routing state, this link must be eliminated from the routing-state result. Based on

this observation we define the response to an equivalent routing state query as below.

[Equivalent Routing-State Response] A response to a declarative equivalence based routing-state query is the one containing all the links whose requested attributes appear in the minimal constraint set.

To describe the routing state response, different output formats are used for different flow descriptions. For those specified by EndpointFilter, we will use the path vector format defined in [I-D.yang-alto-path-vector]. For those specified by FlowFilter, we will use an extended format defined in Section 4.3.1.

An example of the equivalent routing state to a query with the following settings is given in Table 3:

- There are two requested flows the same as in Table 2.
- The variable list is [x, y].
- The X0 is empty
- The link constraint list is [R[0] * x + R[1] * y < bandwidth]

Link	R[0]	R[1]	A[1]/bandwidth
ane1	1	1	100M

Table 3: Abstracted Routing State for Querying Bandwidth

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

[UTE Completeness] Any upper-layer routing (traffic engineering) algorithm where the goal of RSADE in the lower-layer network is to avoid congestion of shared links or shared risk groups can be implemented using the declarative equivalence based routing-state

query. We refer to this as the upper-layer traffic engineering (UTE). Let A = R and b = cap. Then the RSADE query returns a link only if the link may become a bottleneck in the upper layer network.

4. Specification for RSADE Service

In this section we describe the specifications for the RSADE service.

4.1. Information Resource Directory

We introduce the RSADE service as an extension to the Endpoint Cost Service. First we extend the cost type to indicate that this resource supports RSADE service. Second, we add a new capability named "network-attribute-names" to announce the names of network attributes that can be used in the link-constraints. The exact specifications of the corresponding network attributes will be announced in the meta field of an IRD, just like cost types. There are also other drafts proposing extensions on unified properties, such as [I-D.roome-alto-unified-props].

```
{
  "meta": {
    "cost-types: {
      "rsade-cost-type: {
        "cost-metric":  "rsade",
        "cost-mode":    "path-vector",
        "description":  "The cost type for RSADE service"
      }
    },
    "network-attributes": {
      "bandwidth": ...,
      "link-capacity": ...
    }
  },
  "resources": {
    "rsade-example": {
      "uri":        "http://alto.example.com/ecs/rsade",
      "media-type": "application/alto-endpointcost+json",
      "accepts":    "application/alto-endpointcostparams+json",
      "capabilities": {
        "cost-type-names":          [ "rsade-cost-type" ],
        "network-attribute-names":  [ "bandwidth", "link-capacity" ]
      }
    }
  }
}
```

Figure 5: IRD Extensions for RSADE

4.2. Input

As described in Section 3.2.2, applications using the RSADE service should specify flows as well as the equivalence conditions of these flows. In this document, we define the input of RSADE service as RSQuery:

```
object {
  CostType              cost-type;
  [EndpointFilter       endpoints;]
  [FlowFilter           flows;]

  RSEquivCond           equiv-cond;
} RSQuery;

object {
  FlowSpec              specs<1..*>;
} FlowFilter;

object {
  TypedEndpointAddress  src;
  TypedEndpointAddress  dst;
} FlowSpec;

object {
  VariableName          variables<1..*>;
  [SimpleConstraint     basic-constraints<0..*>;]
  LinkConstraint        link-constraints<1..*>;
} RSEquivCond;
```

Figure 6: Specification for the RSQuery

The VariableName should have the same format as the network attribute names, with no more than 32 characters and contains only the ASCII alphanumeric characters (U+0030-U+0039, U+0041-U+005A, and U+0061-U+007A) and the underscore '_'. Also they MUST not start with an ASCII numeric character (U+0030-U+0039).

To avoid collision, the values in the network-attribute-names MUST not appear in the variables field. Otherwise the server SHOULD return an error with the "E_INVALID_FIELD_VALUE" code, as defined in [RFC7285].

Network attribute names MUST no appear in a SimpleConstraint and each LinkConstraint MUST contain at least 1 network attribute name and at least one variable name.

The special network attribute, the routing ratio R, can always be used in a RSADE query and MUST appear as R[?] in a LinkConstraint. The valid index range is determined by the type of the flow-desc. For an EndpointFilter with N srcs and M dsts, the range will be N*M and the routing ratio for the flow between src_i and dst_j is R[(i-1)*M + j] where 1<=i<=N, 1<=j<=M. For an FlowFilter with N flows, it's much more straightforward: R[i] represents the routing ratio for the i-th flow.

4.3. Output

As described in Section 3.2.3, the response to a declarative equivalence based routing-state query should include flow specifications and the corresponding minimal constraints. Different flow specifications have different format of response. For those specified by EndpointFilter, we will use the path vector format defined in [I-D.yang-alto-path-vector]. Here we define the format of response for FlowFilter.

4.3.1. Response for FlowFilter

```
object {
  CostType              cost-type;
  FlowFilter            flows;

  RSEquivCond           min-constraints;
} RSResponse;
```

Figure 7: Specification for the FlowFilter RSResponse

The minimal constraints in RSResponse are generated based on both the equivalence conditions submitted by applications and also the link constraints in the network. The minimal constraints have the same format of RSEquivCond that includes simple-constraints and link-constraints.

5. Security Considerations

This document has not conducted its security analysis.

6. IANA Considerations

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

7. Acknowledgments

The author thanks discussions with Jun Bi and Andreas Voellmy.

8. References

8.1. Normative References

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

8.2. Informative References

[I-D.ietf-alto-incr-update-sse]
Roome, W. and Y. Yang, "ALTO Incremental Updates Using Server-Sent Events (SSE)", draft-ietf-alto-incr-update-sse-02 (work in progress), April 2016.

[I-D.lee-alto-app-net-info-exchange]
Lee, Y., Dhody, D., Wu, Q., Bernstein, G., and T. Choi, "ALTO Extensions to Support Application and Network Resource Information Exchange for High Bandwidth Applications in TE networks", draft-lee-alto-app-net-info-exchange-04 (work in progress), October 2013.

[I-D.roome-alto-unified-props]
Roome, W., "Extensible Property Maps for the ALTO Protocol", draft-roome-alto-unified-props-00 (work in progress), July 2015.

[I-D.yang-alto-path-vector]
Bernstein, G., Gao, K., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Extension: Path Vector Cost Mode", draft-yang-alto-path-vector-03 (work in progress), July 2016.

[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, <http://www.rfc-editor.org/info/rfc7285>.

Authors' Addresses

Kai Gao
Tsinghua University
30 Shuangqinglu Street
Beijing 100084
China

Email: gaok12@mails.tsinghua.edu.cn

Xin (Tony) Wang
Tongji University
4800 CaoAn Road
Shanghai 210000
China

Email: xinwang2014@hotmail.com

Chen Gu
Tongji University
4800 CaoAn Road
Shanghai 210000
China

Email: gc19931011jy@gmail.com

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

Email: yry@cs.yale.edu

G. Robert Chen
Huawei
Nanjing
China

Email: chenguohai@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 8, 2017

S. Hommes
B. Fiz
R. State
University of Luxembourg
A. Zuenko
R. Caetano
Stratumn
V. Gurbani
Bell Laboratories

July 7, 2016

ALTO for the blockchain
draft-hommes-alto-blockchain-01

Abstract

With the inception of the Bitcoin cryptocurrency, the underlying concept of the blockchain has now attracted a large number of application scenarios. Due to the decentralised nature of a typical blockchain service, a reliable communication between the different nodes is a mandatory requirement. RFC7285 describes the idea of using Application-Layer Traffic Optimization (ALTO) that is used to improve the communication in peer-to-peer networks. This document describes the benefits of using ALTO in the context of a blockchain network, and highlights the improvements for a private, consortium and public blockchain.

Status of This Memo

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

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

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

This Internet-Draft will expire on January 8, 2017.

Table of Contents

1. Introduction

With the inception of the Bitcoin cryptocurrency, the underlying concept of the blockchain has now attracted a large number of application scenarios.

The blockchain is a distributed ledger technology that stores assets in the form of transactions. All transactions are further collected into blocks. The process of finding a new block requires the validation of transactions, which are then included into the blocks. Each block contains a chained hash of its previous block, which protects the blockchain against alterations. Any modification of a block would require a recalculation of all subsequent blocks in the blockchain, which is computational very expensive. This property assures an unchangeable history and allows for non-repudiation and a way to track all transactions that have ever occurred on that blockchain.

A typical blockchain service consists of a number of different nodes that communicate via a peer-to-peer protocol in a decentralised network. Each node, which is further considered as a peer, contains a copy of the whole blockchain. In order to maintain the blockchain copies stored across the different nodes synchronised, a blockchain network will have a broadcasting mechanism for new transactions and blocks. Given that these communications can get relatively large in size, in particular in the case of blocks, traditionally the broadcasting mechanism is performed as follows: (1) A node broadcasts its available transactions/blocks to its neighbours. (2) If the neighbouring node does not have the transaction and/or block, it will issue a request to receive it and the transactions and/or blocks will be transferred. (3) If no new transaction/block was offered to the node, the message will be ignored.

ALTO could reduce the number of ignored and therefore redundant broadcasting messages by optimising the network topology of the decentralised network. This could be realised by reducing the connection between nodes based on the topology information that is available to the ALTO server. Moreover, ALTO can provided guidance by selecting the optimal route in order to optimise the network metrics (e.g. latency, bandwidth). ALTO can also provide an information to peers in order to assure Quality-of-service (QoS) requirements.

The following paragraphs highlight some important aspects that improve a network of peers belonging to a blockchain network when using Application-Layer Traffic Optimization (ALTO). We further consider three categories of blockchain networks, which are private, consortium and public.

The ALTO protocol is specified in [RFC7285]. An ALTO server discovery procedure is defined in [RFC7286].

2. Private Blockchains

A private blockchain network is considered as a network that is controlled by a single entity. Such a network is typically located at a certain location, or might be separated at different locations that communicate with each other.

The benefit of using ALTO for deciding on which peer to select is applicable in case of a distributed deployment that is separated over multiple nodes. Privacy issues are of less importance since the blockchain is owned by a single entity that will also provide the ALTO server.

3. Consortium Blockchains

A consortium blockchain network is based on a pre-defined group of participants, for instance a consortium of companies belonging to a certain industry. Each node needs to register before it is allowed to participate in the network, and authentication methods are used to control the access.

The consensus model of this type of blockchain leads typically to an intensive exchange of data between nodes. For instance, the Tendermint developers limit the scale of a core network to 300 nodes due to the excessive communication. The usage of ALTO can help to optimise the communication between peers in order to reduce the communication overhead to a minimum, and increases therefore the number of supported nodes.

4. Public Blockchains

A public blockchain network is considered as a network that is not restricting the access to a certain user group but being available for everyone. Additional registration procedures might be required before joining, but every user is considered as a potential candidate. A typical example for such a network is the Bitcoin network.

Before a new blockchain node can join an existing blockchain service, a bootstrap mechanism is required to select the initial peers that are used to establish the connection. The ALTO server can provide such an information to the nodes before joining the network, or after a node has lost connection and needs to re-connect to the network. Moreover ALTO can advertise peers periodically or in case of an event about alternative routes that would improve the communication.

The bootstrap mechanism is considered as a trust bottleneck in decentralised networks. For private and consortium blockchain networks, the trust is usually given to the operator of the network, and might be additionally secured by a respective Public-Key-Infrastructure (PKI). For such network, the implementation of ALTO is directly related to the trust in the operator and possible without privacy concerns. For a public blockchain network, the use of ALTO should be reduced to a role of guidance on what peers to select. The reason for this is that the operator of ALTO otherwise becomes a central authority in a decentralised network. For instance, the initial peers might be obtained by a trusted source, and ALTO gives a recommendation based on this set of peers by using different kind of metrics.

The usage of ALTO is further considered to enhance the overall communication of peers and the message propagation time. This is especially interesting for public networks, where a large number of peers communicate with each other that are distributed over multiple Autonomous Systems (AS). Network optimisation with ALTO can be realised due to its global view on the network.

5. The Bitcoin Network

An ALTO server can increase the resilience against interrupts of peers and improve the propagation time of messages between peers that are communicating with each other due to its global view on the network. This can further lead to an advantage when using peers that are part of a Bitcoin network.

When a user has executed a transaction by sending bitcoins from his account/wallet to the recipient, the transaction is broadcasted as described in section 1. The Bitcoin network can therefore benefit from ALTO since the number of redundant connections is reduced, which decreases drastically the total amount of traffic in the network.

The usage of ALTO might reduce the occurrence of forks by increasing the propagation time between peers in the network. The process of finding a new block in the Bitcoin network is called mining, and approximately every 10 seconds a new block is added to the blockchain. The mining of new bitcoins requires to solve a well-defined cryptographic problem, the so called proof-of-work. As soon as a new block has been mined, this block is propagated by the network and becomes a potential candidate of a block that is further added to the blockchain as the next block. In case that multiple valid blocks have been found by multiple miners at the same time, a so called fork is generated, resulting in a split of the blockchain into multiple chains that develop independently from each other. As soon as the nodes of a fork are aware of the existence of other forks, the situation is resolved and only a single chain is chosen as valid, meaning that all other forks become invalid including the mined blocks. In order to minimise the occurrence of forks, which provides no reward for the miners that are part of a fork that has been discarded, ALTO can optimise the communication between miners to increase the propagation speed of new blocks.

The Bitcoin network has an additional "relay network", which is a system of high-speed relay nodes that are used as a fall back network for the public Bitcoin network, and to decrease propagation time between miners. An ALTO server could improve the bootstrap process for nodes by proposing the best relay node for a connection.

For the Bitcoin network, which consists of a decentralised network architecture with no single authority, it is important to emphasize that ALTO is considered as a service that can be used to give guidance for the peer selection, but is not replacing the underlying peer-to-peer network. For instance, Bitcoin is currently using a hard-coded list of DNS seeds to obtain the list of peers. ALTO could further be used to exchange information with the DNS seeds, or filtering their response of peers by adding other relevant information obtained by its global view on the network.

6. Role-based Blockchains

Blockchain service providers often have a set of different nodes, where each node has a different role in the network. For instance, the Bitcoin network consists of wallet, miner and relay nodes. Each of them having different requirements in terms of communication preferences. While a wallet node is interested to have as many connection as possible with other wallet nodes, a miner is usually aiming to reduce the propagation delay for transmitting new blocks. An ALTO server can take this into account, and proposes different routes in dependence of the type of node that sent the request. This requires that the ALTO server is aware of the different roles and which metrics need to be applied for each role. The current status of ALTO supports such an application scenario.

7. Security Considerations

From a security perspective, the usage of ALTO in blockchain networks might lead to new kind of attacks. We consider this risk of less importance for a private and consortium network, where all participants are known to the operator and authentication mechanisms are used to restrict access to the network.

For the public blockchain networks, the usage of an ALTO server might lead to new kind of attacks. For instance, an attacker might be able to predict the routes of ALTO and exploit this knowledge in order perform a double spending attack more easily. The scope of such attacks and security violations needs to be investigated and is not part of this draft.

8. Acknowledgments

-

9. Instructions to the RFC Editor

Please remove this section in its entirety before publication as an RFC.

Please replace any instances of RFCxxxx with the RFC number assigned to this memo.

10. Normative References

[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, <http://www.rfc-editor.org/info/rfc7286>.

[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, <http://www.rfc-editor.org/info/rfc7285>.

Authors' Addresses

Stefan Hommes
University of Luxembourg

Email: stefan.hommes@uni.lu

Beltran Fiz
University of Luxembourg

Email: beltran.fiz@uni.lu

Radu State
University of Luxembourg

Email: radu.state@uni.lu

Anton Zuenko
Stratumn

Email: anton@stratumn.com

Richard Caetano
Stratumn

Email: richard@stratumn.com

Vijay Gurbani
Bell Laboratories

Email: vkg@bell-labs.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 15, 2016

S. Randriamasy
W. Roome
Nokia Bell Labs
N. Schwan
Thales Deutschland

June 13, 2016

Multi-Cost ALTO
draft-ietf-alto-multi-cost-02

Abstract

The ALTO (Application Layer-Traffic Optimization) Protocol ([RFC7285]) defines several services that return various metrics describing the costs between network endpoints. For example, when downloading a file that is mirrored on several sites, a user application may use these ALTO cost metrics to determine the most efficient mirror site.

An ALTO Server may offer a variety of cost metrics, based on latency, bandwidth, hop count, jitter, or whatever else the ALTO Server deems useful. When selecting a mirror site, a client may consider more than one metric, perhaps trading bandwidth for latency. While the base ALTO Protocol allows a client to use more than one cost metric, to do so, the client must request each metric separately. This document defines a new service that allows a client to retrieve several cost metrics with one request, which is considerably more efficient. In addition, this document extends the ALTO constraint tests to allow a user to specify an arbitrary logical combination of tests on several cost metrics.

Requirements Language

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

Status of This Memo

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

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

Copyright Notice

Table of Contents

1. Introduction

IETF has designed a new service called ALTO that provides guidance to overlay applications, which have to select one or several hosts from a set of candidates that are able to provide a desired resource. This guidance is based on parameters such as the topological distance, that affect performance and efficiency of the data transmission between the hosts. The purpose of ALTO is to improve Quality of Experience (QoE) in the application while reducing resource consumption in the underlying network infrastructure. The ALTO protocol conveys the Internet View from the perspective of a Provider Network region that spans from a region to one or more Autonomous System (AS) and is called a Network Map. ALTO may also provide the Provider determined Cost Map between locations of the Network Map or Endpoint Cost Map between groups of individual endpoints. Last, these costs are provided as numerical or ordinal values.

Current ALTO Costs and their modes provide values that are seen to be stable over a longer period of time, such as hopcount and administrative routing cost to reflect ISP routing preferences. Recently, new use cases have extended the usage scope of ALTO to Content Delivery Networks (CDN), Data Centers and applications that need additional information to select their Endpoints or handle their PIDs.

Thus a multitude of new Cost Types that better reflect the requirements of these applications are expected to be specified, in particular cost values that change more frequently than previously assumed.

The ALTO protocol [RFC7285] restricts ALTO Cost Maps and Endpoint Cost services to only one Cost Type and Cost Mode per ALTO request.

To retrieve information for several Cost Types, an ALTO client must send several separate requests to the server.

It would be far more efficient, in terms of Round Trip Time (RTT), traffic, and processing load on the ALTO client and server, to get all costs with a single query/response transaction. Vector costs provide a robust and natural input to multi-variate path computation as well as robust multi-variate selection of multiple Endpoints. In particular, one Cost Map reporting on N Cost Types is less bulky than N Cost Maps containing one Cost Type each. This is valuable for both the storage of these maps and for their transmission. Additionally, for many emerging applications that need information on several Cost Types, having them gathered in one map will save time. Another potential advantage is consistency: providing values for several Cost Types in one single batch is useful for Clients needing synchronized ALTO information updates.

Along with multi-cost values queries, the filtering capabilities need to be extended to allow constraints on multiple metrics. The base protocol allows a client to provide optional constraint tests for a Filtered Cost Map or the Endpoint Cost Service. In the base protocol, the constraint tests are limited to the AND-combination of simple comparison tests on the value of the (single) requested Cost Type. It is therefore necessary to allow constraints on multiple metrics. Beyond that, applications that are sensitive to several metrics and struggle with complicated network conditions may need to arbitrate between conflicting objectives such as routing cost and network performance. To address this issue, this document proposes to extend the base protocol by extending constraints to test multiple metrics, and by allowing these constraints to be combined with logical 'ORs' as well as logical 'ANDs'. This allows an application to make requests such as: "select solutions with either (moderate "hopcount" AND high "routingcost") OR (higher "hopcount" AND moderate "routingcost")". To ensure compatibility with legacy ALTO Clients, only the Filtered Cost Map and Endpoint Cost Map services are extended to return Multi-Cost values. Full Cost Map services remain unchanged, and are restricted to returning single cost values.

This document is organized as follows: Section 2 defines terminology used in this document. Section 3 gives a non-normative overview of the multi-cost extensions, and Section 4 gives their formal definitions. Section 5 gives several complete examples. The remaining sections describe the IANA and privacy considerations.

2. Terminology

- {1.2.3}: References with curly brackets are to sections in the ALTO protocol specification [RFC7285].
- Endpoint (EP): A Peer, a CDN storage location, a physical server involved in a virtual server-supported application, a party in a resource sharing swarm such as a computation grid or an online multi-party game.
- Endpoint Discovery (EP Discovery): This term covers the different types of processes used to discover the eligible endpoints.
- Network Service Provider (NSP): Includes both ISPs, who provide means to transport the data, and CDNs who care for the dissemination, persistent storage and possibly identification of the best/closest content copy.
- ALTO transaction: A request/response exchange between an ALTO Client and an ALTO Server.
- Application Client (AC): This term generalizes the case of a P2P client to include the case of a CDN client, a client of an application running on a virtual server, a Grid application client, or any Client that can choose between several connection points for data or resource exchange.

3. Overview Of Approach

The following is a non-normative overview of the multi-cost extensions defined in this document. It assumes the reader is familiar with Cost Map resources in the ALTO Protocol ([RFC7285]).

3.1. Multi-Cost Data Format

Formally, the cost entries in an ALTO Cost Map can be any type of JSON value (see the DstCosts object in {11.2.3.6}). However, that section also says that an implementation may assume costs are JSON numbers, unless the implementation is using an extension which signals a different data type.

Therefore this document extends the definition of a Cost Map to allow a cost to be an array of costs, one per metric, instead of just one number. For example, here is a Cost Map with the "routingcost" and "hopcount" metrics. Note that this is identical to a regular ALTO Cost Map, except that the values are arrays instead of numbers.

```
{
 "meta" : {
   "dependent-vtags" : [ ... ],
   "multi-cost-types" : [
     {"cost-mode": "numerical", "cost-metric": "routingcost"},
     {"cost-mode": "numerical", "cost-metric": "hopcount"}
   ]
 }
 "cost-map" : {
   "PID1": { "PID1":[1,0],  "PID2":[5,23],  "PID3":[10,5] },
   ...
 }
}
```

3.2. Compatibility With Legacy Clients

The multi-cost extensions defined in this document must not break legacy implementations (that is, clients and servers which are not aware of these extensions). One way to achieve that would be to define a new media type for an array-valued Multi Cost Map. However, as indicated above, an array-valued Multi Cost Map is almost identical to a single-valued Cost Map, so it should be simple to write a parser which handles either type of cost map. Hence defining a new media type could result in a lot of wasteful duplication.

Therefore this document does not define any new media types. Instead, as described below, it extends the specifications in the ALTO Server's Information Resource Directory (IRD) so that legacy clients will not request array-valued Multi Cost Map resources. This relies on the requirement that ALTO Clients MUST ignore unknown fields ({8.3.7}).

3.3. Filtered Multi Cost Map Resources

This document extends the Filtered Cost Map service to allow the same resource to return either a single-valued Cost Map, as defined in [RFC7285], or an array-valued Multi Cost Map, as defined in this document. An extended Filtered Cost Map resource has a new capability, "max-cost-types". The value is the maximum number of cost types this resource can return for one request. The existence of this capability means the resource understands the extensions in this document.

For example, the following fragment from an IRD defines an extended Filtered Cost Map resource:

```
    "filtered-multicost-map" : {
      "uri" : "http://alto.example.com/multi/costmap/filtered",
      "media-types" : [ "application/alto-costmap+json" ],
      "accepts" : [ "application/alto-costmapfilter+json" ],
      "uses" : [ "my-default-network-map" ],
      "capabilities" : {
        "max-cost-types" : 2,
        "cost-type-names" : [ "num-routingcost",
                              "num-hopcount" ],
        ...
      }
```

A legacy client will ignore the "max-cost-types" capability, and will send a request with the input parameter "cost-type" describing the desired cost metric, as defined in [RFC7285]. The ALTO Server will return a single-valued legacy Cost Map.

However, a multi-cost-aware client will realize that this resource supports the multi-cost extensions, and can send a POST request with the new input parameter "multi-cost-types", whose value is an array of cost types. Because the request has the "multi-cost-types" parameter (rather than the "cost-type" parameter defined in the base protocol), the server realizes that the client also supports the extensions in this document, and hence responds with a Multi Cost Map, with the costs in the order listed in "multi-cost-types".

3.4. Endpoint Cost Service Resources

This document uses the technique described in Section 3.3 to extend the Endpoint Cost Service to return array-valued costs to clients who also are aware of these extensions.

3.5. Full Cost Map Resources

Full Cost Map resources are GET-mode requests, with no capabilities other than the name of the cost type they return. Therefore unless we create a new media type for array-valued Cost Maps, it is not possible to define a Multi-Cost Full Cost Map resource so that multi-cost-aware clients can recognize it and legacy clients will ignore it. Indeed, the response for a Full Cost Map conveying multiple cost types would include a "meta" field that would itself include a "cost-type" field, that would list several values corresponding to the cost types of the cost map. A legacy client would not be able to understand this list. It would not know what the cost type of the map is and neither would it be able to interpret the cost values array provided by a Multi-Cost full maps.

However {11.3.2.3} of [RFC7285] requires a Filtered Cost Map to return the entire Cost Map if the client omits the source and destination PIDs. Hence a client can use an extended Filtered Cost Map resource to get a full Multi Cost Map.

3.6. Extended Constraint Tests

[RFC7285] defines a simple constraint test capability for Filtered Cost Maps and Endpoint Cost Services. If a resource supports constraints, the server restricts the response to costs that satisfy a list of simple predicates provided by the client. For example, if the client gives the constraints

```
"constraints": ["ge 10", "le 20"]
```

Then the server only returns costs in the range [10,20].

To be useful with multi-cost requests, the constraint tests require several extensions.

3.6.1. Extended constraint predicates

First, because a multi-cost request involves more than one cost metric, the simple predicates must be extended to specify the metric to test. Therefore we extend the predicate syntax to "[##] op value", where "##" is the index of a cost metric in this multi-cost request.

3.6.2. Extended logical combination of predicates

Second, once multiple cost metrics are involved, the "AND" of simple predicates is no longer sufficient. To be useful, clients must be able to express "OR" tests. Hence we add a new field, "or-constraints", to the client request. The value is an array of arrays of simple predicates, and represents the OR of ANDs of those predicates.

Thus, the following request tells the server to limit its response to cost points with "routingcost" <= 100 AND "hopcount" <= 2, OR else "routingcost" <= 10 AND "hopcount" <= 6:

```
{
  "multi-cost-types": [
      {"cost-metric": "routingcost", "cost-mode": "numerical"},
      {"cost-metric": "hopcount",    "cost-mode": "numerical"}
  ],
  "or-constraints": [
      ["[0] le 100", "[1] le 2"],
      ["[0] le 10",  "[1] le 6"]
  ],
  "pids": {...}
}
```

3.6.3. Testable Cost Types in constraints

Finally, a client may want to test a cost type whose actual value is irrelevant, as long as it satisfies the tests. For example, a client may want the value of the cost metric "routingcost" for all PID pairs that satisfy constraints on the metric "hopcount", without needing the actual value of "hopcount".

For example, the following request tells the server to return just "routingcost" for those source and destination pairs for which "hopcount" is <= 6:

```
{
  "multi-cost-types": [
      {"cost-metric": "routingcost", "cost-mode": "numerical"},
  ],
  "testable-cost-types": [
      {"cost-metric": "hopcount", "cost-mode": "numerical"},
  ],
  "constraints": ["[0] le 6"],
  "pids": {...}
}
```

In this example, "[0]" means the constraint applies to "hopcount" because that is the first cost type in the "testable-cost-types" parameter. (If "testable-cost-types" is omitted, it is assumed to be the same as "multi-cost-types".)

3.6.4. Testable Cost Type Names in IRD capabilities

In [RFC7285], when a resource’s capability "constraints" is true, the server accepts constraints on all the cost types listed in the "cost-type-names" capability. However, some ALTO Servers may not be willing to allow contraint tests on all available cost metrics. Therefore the Multi-Cost ALTO protocol extension defines the capability field "testable-cost-type-names". Like "cost-type-names",

it is an array of cost type names. If present, that resource only allows constraint tests on the cost types in that list. "testable-cost-type-names" MUST be a subset of "cost-type-names".

3.6.5. Legacy client issues

While a multi-cost-aware client will recognize the "testable-cost-type-names" field, and will honor those restrictions, a legacy client will not. Hence a legacy may send a request with a constraint test on any of the cost types listed in "cost-type-names".

To avoid that problem, the "testable-cost-type-names" and "cost-constraints" fields are mutually exclusive: a resource may define one or the other capability, but MUST NOT define both. Thus a resource that does not allow constraint tests on all cost metrics will set "testable-cost-type-names" to the testable metrics, and will set "cost-constraints" to "false". A multi-cost-aware client will recognize the "testable-cost-type-names" field, and will realize that its existence means the resource does allow (limited) contraint tests, while a legacy client will think that resource does not allow constraint tests at all. To allow legacy clients to use constraint tests, the ALTO Server MAY define an additional resource with "cost-constraints" set to "true" and "cost-type-names" set to the metrics which can be tested.

In the IRD example below, the resource "filtered-cost-map-extended" provides values for three metrics: "num-routingcost", "num-hopcount" and "num-bwscore". The capability "testable-cost-type-names" indicates that the server only allows constraints on "routingcost" and "hopcount". A multi-cost capable client will see this capability, and will limit its constraint tests to those metrics. Because capability "cost-constraints" is false (by default), a legacy client will not use constraint tests on this resource at all.

The second resource, "filtered-multicost-map", is similar to the first, except that all the metrics it returns are testable. Therefore it sets "cost-constraints" to "true", and does not set the "testable-cost-type-names" field. A legacy client that needs a constraint test will use this resource rather than the first. A multi-cost-aware client that does not need to retrieve the "num-bwscore" metric may use either resource.

```
   "filtered-cost-map-extended" : {
      "uri" : "http://alto.example.com/multi/extn/costmap/filtered",
      "media-types" : [ "application/alto-costmap+json" ],
      "accepts" : [ "application/alto-costmapfilter+json" ],
      "uses" : [ "my-default-network-map" ],
      "capabilities" : {
         "max-cost-types" : 3,
         "cost-type-names" : [ "num-routingcost",
                               "num-hopcount",
                               "num-bwscore"],
         "testable-cost-type-names" : [ "num-routingcost",
                                        "num-hopcount" ]
      }
   },

   "filtered-multicost-map" : {
      "uri" : "http://alto.example.com/multi/costmap/filtered",
      "media-types" : [ "application/alto-costmap+json" ],
      "accepts" : [ "application/alto-costmapfilter+json" ],
      "uses" : [ "my-default-network-map" ],
      "capabilities" : {
        "cost-constraints" : true,
        "max-cost-types" : 2,
        "cost-type-names" : [ "num-routingcost",
                              "num-hopcount"],
      }
   }
```

4. Protocol Extensions for Multi-Cost ALTO Transactions

This section formally specifies the extensions to [RFC7285] to support Multi-Cost ALTO transactions.

4.1. Filtered Cost Map Extensions

This document extends Filtered Cost Maps, as defined in {11.3.2} of [RFC7285], by adding new input parameters and capabilities, and by returning JSONArrays instead of JSONNumbers as the cost values.

The media type (11.3.2.1}, HTTP method (11.3.2.2} and "uses" specifications (11.3.2.5} are unchanged.

4.1.1. Capabilities

The filtered cost map capabilities are extended with two new members:

o max-cost-types,

o testable-cost-type-names

The capability "max-cost-types" indicates whether this resource supports the Multi-Cost ALTO extensions, and the capability "testable-cost-type-names" allows the resource to restrict constraint tests to a subset of the available cost types. The FilteredCostMapCapabilities object in {11.3.2.4} is extended as follows:

```
object {
   JSONString cost-type-names<1..*>;
   [JSONBool cost-constraints;]
   [JSONNumber max-cost-types;]
   [JSONString testable-cost-type-names<1..*>;]
} FilteredCostMapCapabilities;
```

cost-type-names and cost-constraints: As defined in {11.3.2.4} of [RFC7285].

max-cost-types: 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.

testable-cost-type-names: If present, the resource allows constraint tests, but only on the cost type names in this array. Each name in "testable-cost-type-names" MUST also be in "cost-type-names". If "testable-cost-type-names" is present, the "cost-constraints" capability MUST NOT be "true", and if "cost-constraints" is "true", "testable-cost-type-names" MUST NOT be present. Thus if "cost-constraints" is "true", the resource MUST accept constraint tests on any cost type in "cost-type-names".

As discussed in Section 3.6.4, this capability is useful when a server is unable or unwilling to implement constraint tests on all cost types. As discussed in Section 3.6.5, "testable-cost-type-names" and "cost-constraints" are mutually exclusive to prevent legacy clients from issuing constraint tests on untestable cost types.

4.1.2. Accept Input Parameters

The ReqFilteredCostMap object in {11.3.2.3} of [RFC7285] is extended as follows:

```
object {
   [CostType cost-type;]
   [CostType multi-cost-types<1..*>;]
   [CostType testable-cost-types<1..*>;]
   [JSONString constraints<0..*>;]
   [JSONString or-constraints<0..*><0..*>;]
   PIDFilter pids;
} ReqFilteredCostMap;

object {
   PIDName srcs<0..*>;
   PIDName dsts<0..*>;
} PIDFilter;
```

cost-type: As defined in {11.3.2.3} of [RFC7285], with the additional requirement that the client MUST specify either "cost-type" or "multi-cost-types", but MUST NOT specify both.

multi-cost-types: If present, the ALTO Server MUST return array-valued costs for the cost types in this list. For each entry, the "cost-metric" and "cost-mode" fields MUST match one of the supported cost types indicated in member "cost-type-names" of this resource's "capabilities" field (Section 4.1.1). The client MUST NOT use this field unless this resource's "max-cost-types" capability exists and has a value greater than 0. This field MUST NOT have more than "max-cost-types" cost types. The client MUST specify either "cost-type" or "multi-cost-types", but MUST NOT specify both.

Note that if "multi-cost-types" has one cost type, the values in the cost map will be arrays with one value.

testable-cost-types: A list of cost types used for extended constraint tests, as described for the "constraints" and "or-constraints" parameters. These cost types must either be a subset of the cost types in the resource's "testable-cost-type-names" capability (Section 4.1.1), or else, if the resource's capability "cost-constraints" is true, a subset of the cost types in the resource's "cost-type-names" capability

If "testable-cost-types" is omitted, it is assumed to have the cost types in "multi-cost-types" or "cost-type".

This feature is useful when a client wants to test a cost type whose actual value is irrelevant, as long as it satisfies the tests. For example, a client may want the cost metric

"routingcost" for those PID pairs whose "hopcount" is less than 10. The exact hopcount does not matter.

constraints: If this resource's "max-cost-types" capability (Section 4.1.1) has the value 0 (or is not defined), this parameter is as defined in {11.3.2.3} of [RFC7285]: an array of constraint tests related to each other by a logical AND. In this case it MUST NOT be specified unless the resource's "cost-constraints" capability is "true".

If this resource's "max-cost-types" capability has a value greater than 0, then a "constraints" parameter with the array of extended predicates [P1, P2, ...] is equivalent to an "or-constraints" parameter with the value [[P1, P2, ...]]. In this case, the "constraints" parameter MUST NOT be specified if the "or-constraints" parameter is specified.

or-constraints: A JSONArray of JSONArrays of JSONStrings, where each string is an extended constraint predicate as defined below. The "or-constraint" tests are interpreted as the logical OR of ANDs of predicates. That is, the ALTO Server should return a cost point only if it satisfies all constraints in any one of the sub-arrays.

This parameter MAY be specified if this resource's "max-cost-types" capability is defined with a value greater than 0 (Section 4.1.1), and if the resource allows constraint tests (the resource's "cost-constraints" capability is "true" or its "testable-cost-type-names" capability is not empty). Otherwise this parameter MUST NOT be specified.

This parameter MUST NOT be specified if the "constraints" parameter is specified.

An extended constraint predicate consists of two or three entities separated by white space: (1) an optional cost type index, of the form "[#]", with default value "[0]", (2) a required operator, and (3) a required target value. The operator and target value are as defined in {11.3.2.3} of [RFC7285]. The cost type index, i, specifies the cost type to test. If the "testable-cost-type" parameter is present, the test applies to the i'th cost type in "testable-cost-types", starting with index 0. Otherwise if the "multi-cost-types" parameter is present, the test applies to the i'th cost type in that array. If neither parameters are present, the test applies to the cost type in the "cost-type" parameter, in which this case the index MUST be 0. Regardless of how the tested cost type is selected, it MUST be in the resource's "testable-cost-type-names" capability, or, if not present, in the "cost-type-names" capability.

As an example, suppose "multi-cost-types" has the single element "routingcost", "testable-cost-types" has the single element "hopcount", and "or-constraints" has the single element "[0] le 5". This is equivalent to the database query "SELECT and provide routingcost WHERE hopcount <= 5".

Note that the index is optional, so a constraint test as defined in {11.3.2.3}, such as "le 10", is equivalent to "[0] le 10". Thus legacy constraint tests are also legal extended constraint tests.

Also note that if the "max-cost-types" capability has a value greater than 0, a client MAY use the "or-constraints" parameter together with the "cost-type" parameter. That is, if the client and server are both aware of the extensions in this document, a client MAY use an "OR" test for a single-valued cost request.

pids, srcs, dsts: As defined in {11.3.2.3} of [RFC7285].

4.1.3. Response

If the client specifies the "cost-type" input parameter, the response is exactly as defined in {11.2.3.6} of [RFC7285]. If the client provides the "multi-cost-types" instead, then the response is changed as follows:

- In "meta", the field "cost-type" is replaced with the field "multi-cost-types", with the same value as the "multi-cost-types" input parameter.

- The costs are JSONArrays, instead of JSONNumbers. All arrays have the same cardinality as the "multi-cost-types" input parameter, and contain the cost type values in that order. If a cost type is not available for a particular source and destination, the ALTO Server MUST use the JSON "null" value for that array element. If none of the cost types are available for a particular source and destination, the ALTO Server MAY omit the entry for that source and destination.

4.2. Endpoint Cost Service Extensions

This document extends the Endpoint Cost Service, as defined in {11.5.1} of [RFC7285], by adding new input parameters and capabilities, and by returning JSONArrays instead of JSONNumbers as the cost values.

The media type (11.5.1.1}, HTTP method (11.5.1.2} and "uses" specifications (11.5.1.5} are unchanged.

4.2.1. Capabilities

The extensions to the Endpoint Cost Service capabilities are identical to the extensions to the Filtered Cost Map (see Section 4.1.1).

4.2.2. Accept Input Parameters

The ReqEndpointCostMap object in {11.5.1.3} of [RFC7285] is extended as follows:

```
object {
   [CostType cost-type;]
   [CostType multi-cost-types<1..*>;]
   [CostType testable-cost-types<1..*>;]
   [JSONString constraints<0..*>;]
   [JSONString or-constraints<0..*><0..*>;]
   EndpointFilter endpoints;
} ReqFilteredCostMap;

object {
   [TypedEndpointAddr srcs<0..*>;]
   [TypedEndpointAddr dsts<0..*>;]
} EndpointFilter;
```

cost-type: As defined in {11.5.1.3} of [RFC7285], with the additional requirement that the client MUST specify either "cost-type" or "multi-cost-types", but MUST NOT specify both.

multi-cost-types: If present, the ALTO Server MUST return array-valued costs for the cost types in this list. For each entry, the "cost-metric" and "cost-mode" fields MUST match one of the supported cost types indicated in this resource's "capabilities" field (Section 4.2.1). The client MUST NOT use this field unless this resource's "max-cost-types" capability exists and has a value greater than 0. This field MUST NOT have more than "max-cost-types" cost types. The client MUST specify either "cost-type" or "multi-cost-types", but MUST NOT specify both.

Note that if "multi-cost-types" has one cost type, the values in the cost map will be arrays with one value.

testable-cost-types, constraints, or-constraints: Defined equivalently to the corresponding input parameters for an extended Filtered Cost Map (Section 4.1.2).

endpoints, srcs, dsts: As defined in {11.5.1.3} of [RFC7285].

4.2.3. Response

The extensions to the Endpoint Cost Service response are similar to the extensions to the Filtered Cost Map response (Section 4.1.3). Specifically, if the client specifies the "cost-type" input parameter, the response is exactly as defined in {11.5.1.6} of [RFC7285]. If the client provides the "multi-cost-types" instead, then the response is changed as follows:

- In "meta", the field "cost-type" is replaced with the field "multi-cost-types", with the same value as the "multi-cost-types" input parameter.

- The costs are JSONArrays, instead of JSONNumbers. All arrays have the same cardinality as the "multi-cost-types" input parameter, and contain the cost type values in that order. If a cost type is not available for a particular source and destination, the ALTO Server MUST use the JSON "null" value for that array element. If none of the cost types are available for a particular source and destination, the ALTO Server MAY omit the entry for that source and destination.

5. Examples

5.1. Information Resource Directory

The following is an example of an ALTO Server's Information Resource Directory. In addition to Network and Cost Map resources, it defines two Filtered Cost Map and an Endpoint Cost Service, which all understand the multi-cost extensions.

```
GET /directory HTTP/1.1
Host: alto.example.com
Accept: application/alto-directory+json,application/alto-error+json


HTTP/1.1 200 OK
Content-Length: [TODO]
Content-Type: application/alto-directory+json

{
  "meta" : {
    "default-alto-network-map" : "my-default-network-map",
    "cost-types" : {
       "num-routing" : {
         "cost-mode" : "numerical",
         "cost-metric" : "routingcost"
       },
```

```
          "num-hopcount" : {
            "cost-mode" : "numerical",
            "cost-metric" : "hopcount"
          },
          "num-bwscore" : {
            "cost-mode" : "numerical",
            "cost-metric" : "bandwidthscore"
          },
            .....
            Other ALTO cost types as described in RFC7285
            .....
          }
      },
      "resources" : {
        "my-default-network-map" : {
          "uri" : "http://alto.example.com/networkmap",
          "media-type" : "application/alto-networkmap+json"
        },
        "numerical-routing-cost-map" : {
          "uri" : "http://alto.example.com/costmap/num-routing",
          "media-types" : [ "application/alto-costmap+json" ],
          "uses" : [ "my-default-network-map" ],
          "capabilities" : {
            "cost-type-names" : [ "num-routing" ]
          }
        },
        "numerical-hopcount-cost-map" : {
          "uri" : "http://alto.example.com/costmap/num-hopcount",
          "media-types" : [ "application/alto-costmap+json" ],
          "uses" : [ "my-default-network-map" ],
          "capabilities" : {
            "cost-type-names" : [ "num-hopcount" ]
          }
        },
        .........
        Other resources as described in RFC7285
        .........
        "filtered-multicost-map" : {
          "uri" : "http://alto.example.com/multi/costmap/filtered",
          "media-types" : [ "application/alto-costmap+json" ],
          "accepts" : [ "application/alto-costmapfilter+json" ],
          "uses" : [ "my-default-network-map" ],
          "capabilities" : {
            "cost-constraints" : true,
            "max-cost-types" : 2,
            "cost-type-names" : [ "num-routingcost",
                                  "num-hopcount" ]
          }
```

```
      },
      "filtered-cost-map-extended" : {
        "uri" : "http://alto.example.com/multi/extn/costmap/filtered",
        "media-types" : [ "application/alto-costmap+json" ],
        "accepts" : [ "application/alto-costmapfilter+json" ],
        "uses" : [ "my-default-network-map" ],
        "capabilities" : {
          "max-cost-types" : 3,
          "cost-type-names" : [ "num-routingcost",
                                "num-hopcount",
                                "num-bwscore"],
          "testable-cost-type-names" : [ "num-routingcost",
                                         "num-hopcount" ]
        }
      },
      "endpoint-multicost-map" : {
        "uri" : "http://alto.example.com/multi/endpointcost/lookup",
        "media-types" : [ "application/alto-endpointcost+json" ],
        "accepts" : [ "application/alto-endpointcostparams+json" ],
        "uses" : [ "my-default-network-map" ],
        "capabilities" : {
          "cost-constraints" : true,
          "max-cost-types" : 2,
          "cost-type-names" : [ "num-routingcost",
                                "num-hopcount" ]
        }
      }
    }
  }
```

5.2. Multi-Cost Filtered Cost Map: Example #1

This example illustrates a simple multi-cost ALTO transaction. The ALTO Server provides two Cost Types, "routingcost" and "hopcount", both in "numerical" mode. The ALTO Server does not know the value of the "routingcost" between PID2 and PID3, and hence uses "null" for those costs.

```
POST /multi/costmap/filtered" HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,application/alto-error+json
Content-Type: application/alto-costmapfilter+json
Content-Length: ###

{
  "multi-cost-types": [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "hopcount"}
  ],
  "pids" : {
    "srcs" : [ ],
    "dsts" : [ ]
  }
}


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

{
 "meta" : {
   "dependent-vtags" : [
     {"resource-id": "my-default-network-map",
      "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
     }
   ],
   "multi-cost-types" : [
     {"cost-mode": "numerical", "cost-metric": "routingcost"},
     {"cost-mode": "numerical", "cost-metric": "hopcount"}
   ]
  }
 }
 "cost-map" : {
   "PID1": { "PID1":[1,0],   "PID2":[4,3],    "PID3":[10,2]   },
   "PID2": { "PID1":[15,5],  "PID2":[1,0],    "PID3":[null,9] },
   "PID3": { "PID1":[20,12], "PID2":[null,1], "PID3":[1,0]    }
 }
}
```

5.3. Multi-Cost Filtered Cost Map: Example #2

This example uses constraints to restrict the returned source/ destination PID pairs to those with "routingcost" between 5 and 10, or "hopcount" equal to 0.

```
POST /multi/costmap/filtered HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,application/alto-error+json
Content-Type: application/alto-costmapfilter+json
Content-Length: ###

{
  "multi-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "hopcount"}
  ],
  "or-constraints" : [ ["[0] ge 5", "[0] le 10"],
                       ["[1] eq 0"] ]
  "pids" : {
    "srcs" : [ "PID1", "PID2" ],
    "dsts" : [ "PID1", "PID2", "PID3" ]
  }
}


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

{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
      }
    ],
    "multi-cost-types" : [
      {"cost-mode": "numerical", "cost-metric": "routingcost"},
      {"cost-mode": "numerical", "cost-metric": "hopcount"}
    ]
  }
  "cost-map" : {
    "PID1": { "PID1": [1,0], "PID3": [10,5] },
    "PID2": { "PID2": [1,0]                 }
  }
}
```

5.4. Multi-Cost Filtered Cost Map: Example #3

This example uses extended constraints to limit the response to cost points with ("routingcost" <= 10 and "hopcount" <= 2), or else ("routingcost" <= 3 and "hopcount" <= 6). Unlike the previous

example, the client is only interested in the "routingcost" cost type, and uses the "cost-type" parameter instead of "multi-cost-types" to tell the server to return scalar costs instead of array costs:

```
POST /multi/multicostmap/filtered HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,application/alto-error+json
Content-Type: application/alto-costmapfilter+json
Content-Length: ###

{
  "cost-type" : {
    "cost-mode": "numerical", "cost-metric": "routingcost"
  },
  "testable-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "hopcount"}
  ],
  "or-constraints": [
        ["[0] le 10", "[1] le 2"],
        ["[0] le 3",  "[1] le 6"]
  ],
  "pids" : {
    "srcs" : [ ],
    "dsts" : [ ]
  }
}
```

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

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

5.5. Multi-Cost Filtered Cost Map: Example #4

This example uses extended constraints to limit the response to cost points with ("routingcost" <= 10 and "hopcount" <= 2), or else ("routingcost" <= 3 and "hopcount" <= 6). In this example, the client is interested in the "routingcost" and "bandwidthscore" cost metrics, but not in the "hopcount" metric:

```
POST /multi/extn/costmap/filtered HTTP/1.1
Host: alto.example.com
Accept: application/alto-costmap+json,application/alto-error+json
Content-Type: application/alto-costmapfilter+json
Content-Length: ###

{
  "multi-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "bandwidthscore"}
  ],
  "testable-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "hopcount"}
  ],
  "or-constraints": [
        ["[0] le 10", "[1] le 2"],
        ["[0] le 3",  "[1] le 6"]
  ],
  "pids" : {
    "srcs" : [ ],
    "dsts" : [ ]
  }
}


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

{
  "meta" : {
    "dependent-vtags" : [
      {"resource-id": "my-default-network-map",
       "tag": "3ee2cb7e8d63d9fab71b9b34cbf764436315542e"
      }
    ],
    "multi-cost-types" : [
      {"cost-mode": "numerical", "cost-metric": "routingcost"},
      {"cost-mode": "numerical", "cost-metric": "bandwidthscore"}
    ]
  }
  "cost-map" : {
    "PID1": { "PID1": [1,16] "PID3": [10,19] },
    "PID2": { "PID2": [1,8] },
    "PID3": { "PID3": [1,19] }
  }
}
```

5.6. Endpoint Cost Service

This example uses the Endpoint Cost Service to retrieve the "routingcost" and "hopcount" for selected endpoints, limiting the response to costs with either low hopcount and reasonable routingcost (hopcount <= 2 and routingcost <= 10), or else low routingcost and reasonable hopcount (routingcost <= 3 and hopcount <= 6).

```
POST /multi/endpointcost/lookup HTTP/1.1
Host: alto.example.com
Accept: application/alto-endpointcost+json,
        application/alto-error+json
Content-Type: application/alto-endpoincostparams+json
Content-Length: ###

{
  "multi-cost-types" : [
    {"cost-mode": "numerical", "cost-metric": "routingcost"},
    {"cost-mode": "numerical", "cost-metric": "hopcount"}
  ],
  "or-constraints": [
        ["[0] le 10", "[1] le 2"],
        ["[0] le 3",  "[1] le 6"]
  ],
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2", "ipv6:2001:db8::1:0 ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv4:203.0.113.45",
      "ipv6:2001:db8::10"
    ]
  }
}


HTTP/1.1 200 OK
Content-Length: ###
Content-Type: application/alto-endpointcost+json

{
  "meta" : {
    "multi-cost-types" : [
      {"cost-mode": "numerical", "cost-metric": "routingcost"},
      {"cost-mode": "numerical", "cost-metric": "hopcount"}
    ]
  }
  "endpoint-cost-map" : {
```

```
      "ipv4:192.0.2.2": {
        "ipv4:192.0.2.89":    [15, 5],
        "ipv4:203.0.113.45":  [4, 23]
      }
      "ipv6:2001:db8::1:0": {
        "ipv4:198.51.100.34": [16, 5],
        "ipv6:2001:db8::10":  [10, 2]
      }
    }
  }
```

6. IANA Considerations

This document does define any new media types or introduce any new IANA considerations.

7. Privacy And Security Considerations

This document does introduce any privacy or security issues not already present in the ALTO protocol.

8. Acknowledgements

The authors would like to thank Richard Alimi, Fred Baker, Dhruv Dhodi, Vijay Gurbani, Gao Kai, Dave Mac Dysan, Young Lee, Richard Yang, Qiao Xiang and Wang Xin for fruitful discussions and feedback on this document and previous versions.

9. References

9.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

[RFC5693] "Application Layer Traffic Optimization (ALTO) Problem Statement", October 2009.

[RFC7285] Almi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.

9.2. Informative References

[RFC6708] "Application-Layer Traffic Optimization (ALTO) Requirements", February 2012.

Authors' Addresses

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
NOZAY 91460
FRANCE

Email: Sabine.Randriamasy@nokia-bell-labs.com

Wendy Roome
Nokia Bell Labs
600 Mountain Ave, Rm 3B-324
Murray Hill, NJ 07974
USA

Phone: +1-908-582-7974
Email: w.roome@nokia-bell-labs.com

Nico Schwan
Thales Deutschland
Lorenzstrasse 10
Stuttgart 70435
Germany

Email: nico.schwan@thalesgroup.com

ALTO
Internet-Draft
Intended status: Informational
Expires: January 8, 2017

S. Kiesel
University of Stuttgart
M. Stiemerling
H-DA

July 7, 2016

Application Layer Traffic Optimization (ALTO) Cross-Domain Server Discovery
draft-kiesel-alto-xdom-disc-02

Abstract

The goal of Application-Layer Traffic Optimization (ALTO) is to provide guidance to applications that have to select one or several hosts from a set of candidates capable of providing a desired resource. ALTO is realized by a client-server protocol. Before an ALTO client can ask for guidance it needs to discover one or more ALTO servers that can provide suitable guidance.

In some deployment scenarios, in particular if the information about the network topology is partitioned and distributed over several ALTO servers, it may be needed to discover an ALTO server outside of the own network domain, in order to get appropriate guidance. This document details applicable scenarios, itemizes requirements, and specifies a procedure for ALTO cross-domain server discovery.

Technically, the algorithm specified in this document takes one IP address and a U-NAPTR Service Parameter (i.e., "ALTO:http" or "ALTO:https") as parameters. It performs DNS lookups (for NAPTR resource records in the in-addr.arpa. or ip6.arpa. tree) and returns one or more URI(s) of information resources related to that IP address.

Terminology and Requirements Language

This document makes use of the ALTO terminology defined in RFC 5693 [RFC5693].

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

Status of this Memo

Copyright Notice

Table of Contents

1. Introduction

The goal of Application-Layer Traffic Optimization (ALTO) is to provide guidance to applications that have to select one or several hosts from a set of candidates capable of providing a desired resource [RFC5693]. ALTO is realized by an HTTP-based client-server protocol [RFC7285], which can be used in various deployment scenarios [I-D.ietf-alto-deployments].

1.1. Multiple Information Sources and Partitioned Knowledge

The ALTO base protocol document [RFC7285] specifies the communication between an ALTO client and a single ALTO server. It is implicitly assumed that this server can answer any query, possibly with some kind of default value if no exact data is known. No special provisions were made for the case that the ALTO information originates from multiple sources, which are possibly under the control of different administrative entities (e.g., different ISPs) or that the overall ALTO information is partitioned and stored on several ALTO servers.

1.1.1. Classification of Solution Approaches

Various protocol extensions and other solutions have been proposed to deal with multiple information sources and partitioned knowledge. They can be classified as follows:

1 Ensure that all ALTO servers have the same knowlegde

1.1 Ensure data replication and synchronization within the provisioning protocol (cf. RFC 5693, Fig 1 [RFC5693]).

1.2 Use an Inter-ALTO-server data replication protocol. Possibly, the ALTO protocol itself - maybe with some extensions - could be used for that purpose; however, this has not been studied in detail so far.

2 Accept that different ALTO servers (possibly operated by different organizations, e.g., ISPs) do not have the same knowledge

2.1 Allow ALTO clients to send arbitrary queries to any ALTO server (e.g. the one discovered using [RFC7286]). If this server cannot answer the query itself, it will fetch the data on behalf of the client, using the ALTO protocol or a to-be-defined inter-ALTO-server request forwarding protocol.

2.2 Allow ALTO clients to send arbitrary queries to any ALTO server (e.g. the one discovered using [RFC7286]). If this server cannot answer the query itself, it will redirect the client to the "right" ALTO server that has the desired information, using a small to-be-defined extension of the ALTO protocol.

2.3 ALTO clients need to use some kind of "search engine" that indexes ALTO servers and redirects and/or gives cached results.

2.4 ALTO clients need to use a new discovery mechanism to discover the ALTO server that has the desired information and contact it directly.

1.1.2. Discussion of Solution Approaches

The provisioning or initialization protocol for ALTO servers (cf. RFC 5693, Fig 1 [RFC5693]) is currently not standardized. It was a conscious decision not to include this in the scope of the IETF ALTO working group. The reason is that there are many different kinds of information sources. This implementation specific protocol will adapt them to the ALTO server, which offers a standardized protocol to the ALTO clients. However, adding the task of synchronization between ALTO servers to this protocol (i.e., approach 1.1) would overload this protocol with a second functionality that requires standardization for seamless multi-domain operation.

For the 1.? solution approaches, in addition to general technical feasibility and issues like overhead and caching efficiency, another aspect to consider is legal liability. Operator "A" might prefer not to publish information about nodes in or paths between the networks of operators "B" and "C" through A's ALTO server, even if A knew that information. This is not only a question of map size and processing load on A's ALTO server. Operator A could also face legal liability issues if that information had a bad impact on the traffic engineering between B's and C's networks, or on their business models.

No specific actions to build a "search engine" based solution (approach 2.3) are currently known and it is unclear what could be the incentives to operate such an engine. Therefore, this approach is not considered in the remainder of this document.

1.2. The Need for Cross-Domain ALTO Server Discovery

Approaches 1.1, 1.2, 2.1, and 2.2 do not only require the specification of an ALTO protocol extension or a new protocol that runs between ALTO servers. A large-scale, maybe Internet-wide, multi-domain deployment would also need mechanisms by which an ALTO

server could discover other ALTO servers, learn which information is available where, and ideally also who is authorized to publish information related to a given part of the network. Approach 2.4 needs the same mechanisms, except that they are used on the client-side instead of the server-side.

It is sometimes questioned whether there is a need for a solution that allows clients to ask arbitrary queries, even if the ALTO information is partitioned and stored on many ALTO servers. The main argument is, that clients are supposed to optimize the traffic from and to themselves, and that the information needed for that is most likely stored on a "nearby" ALTO server, i.e., the one that can be discovered using [RFC7286]. However, there are scenarios where the ALTO client is not co-located with an endpoint of the to-be-optimized data transmission. Instead, the ALTO client is located at a third party, which takes part in the application signaling, e.g., a so-called "tracker" in a peer-to-peer application. One such scenario, where it is advantageous to place the ALTO client not at an endpoint of the user data transmission, is analyzed in Appendix B.

1.3. Solution Approach

Several solution approaches for cross-domain ALTO server discovery have been evaluated, using the criteria documented in Appendix A. One of them was to use the ALTO protocol itself for the exchange of information availability [I-D.kiesel-alto-alto4alto]. However, the drawback of that approach is that a new registration administration authority would have to be established.

This document specifies a DNS-based procedure for cross-domain ALTO server discovery, which was inspired by "Location Information Server (LIS) Discovery Using IP Addresses and Reverse DNS" [RFC7216]. The primary goal is that this procedure can be used on the client-side (i.e., approach 2.4), but together with new protocols or protocol extensions it could also be used to implement the other solution approaches itemized above.

1.4. ALTO Requirements

During the design phase of the overall ALTO solution, two different server discovery scenarios have been identified and documented in the ALTO requirements document [RFC6708]. The first scenario, documented in Req. AR-32, can be supported using the discovery mechanisms specified in [RFC7286]. An alternative approach, based on IP anycast [I-D.kiesel-alto-ip-based-srv-disc], has also been studied. This document, in contrast, tries to address Req. AR-33.

1.5. Document History

This document is a direct successor of [I-D.kiesel-alto-3pdisc] and [I-D.kist-alto-3pdisc]. The scenario and mechanisms described here and in these documents have been referred to as "third-party server discovery" in the past. However, to avoid naming ambiguities with a completely different scenario, it has been renamed to "ALTO Cross-Domain Server Discovery".

1.6. Feedback

Comments and discussions about this document should be directed to the ALTO working group: alto@ietf.org.

2. ALTO Cross-Domain Server Discovery Procedure Specification

2.1. Interface

The algorithm specified in this document takes one IP address and a U-NAPTR [RFC4848] Service Parameter (i.e., "ALTO:http" or "ALTO:https") as parameters. It performs DNS lookups (for NAPTR resource records) and returns one or more URI(s) of information resources related to that IP address.

2.2. Basic Principle

This algorithm closely follows [RFC7216] and re-uses parts of [RFC7286].

The algorithm sequentially tries two different lookup strategies. First, an ALTO-specific U-NAPTR record is searched in the "reverse tree", i.e., in subdomains of in-addr.arpa. or ip6.arpa. corresponding to the given IP address. If this lookup does not yield a usable result, further lookups with truncated domain names may be tried. The goal is to allow deployment scenarios that require fine-grained discovery on a per-IP basis, as well as large-scale scenarios where discovery is to be enabled for a large number of IP addresses with a small number of additional DNS resource records.

2.3. Step 1: Prepare Domain Name for Reverse DNS Lookup

This task takes the IP address parameter the procedure was called with and constructs a domain name, which is used for DNS lookups in subsequent tasks.

If the IP address given as a parameter to the procedure is an IPv4 address, the domain name is constructed according to the rules specified in Section 3.5 of [RFC1035] and it is rooted in the in the special domain "IN-ADDR.ARPA.". For IPv6 addresses, the construction rules in Section 2.5 of [RFC3596] apply and the special domain "IP6.ARPA." is used.

Example values for IPv4 and IPv6 addresses could be (Note: a line break was added in the IPv6 example):

```
R:="3.100.51.198.in-addr.arpa."

R:="0.2.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.0.8.B.D.0.
1.0.0.2.ip6.arpa."
```

2.4. Step 2: Add Shortened Domain Names

This task creates a list of several additional domain names, based on the domain name yielded in Step 1.

- For IP version 4, the domain name from Step 1 SHOULD be shortened successively by one and two labels (i.e., purge the first or second dot from the left and everything left of it, respectively), and the results being added to the list. This corresponds to a search on a /24 or /16 network prefix.

- For IP version 6, the domain name from Step 1 SHOULD be shortened successively by 16, 18, 20, and 24 labels, and the results being added to the list. This corresponds to a search on a /64, /56, /48, or /32 network prefix.

This list is intended to provide network operators with a degree of flexibility in where discovery-related resource records can be placed without significantly increasing the number of DNS names that are searched. This does not attach any other significance to these specific zone cuts or create a classful addressing hierarchy based on the reverse DNS tree.

For example, the IPv4 address "192.0.2.75" could result in a list of domain names (with the result from Step 1 put in the first position):

- 75.2.0.192.in-addr.arpa.

- 2.0.192.in-addr.arpa.

- 0.192.in-addr.arpa.

Similarly, the IPv6 address "2001:DB8::28e4:3a93:4429:dfb5" could result in a list:

- 5.b.f.d.9.2.4.4.3.9.a.3.4.e.8.2.0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.

- 0.0.0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.

- 0.0.0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.

- 0.0.0.0.8.b.d.0.1.0.0.2.ip6.arpa.

- 8.b.d.0.1.0.0.2.ip6.arpa.

The limited number of labels by which each name is shortened is intended to limit the maximum number of DNS queries produced by a

single invocation of the cross-domain ALTO server discovery procedure. No more than five U-NAPTR resolutions are invoked for each IP address.

2.5. Step 3: DNS lookups

The list of domain names which was created in the previous step is sequentially (from longest to shortest name) processed, as described in Section 3.2 of RFC 7286 [RFC7286].

3. Using ALTO Cross-Domain Server Discovery with the ALTO Protocol

TBD: expand

3.1. Endpoint Property Service

If an ALTO client wants to query the Endpoint Property Service (see Section 11.4 of RFC 7285 [RFC7285]) for an endpoint with IP address X, it has to invoke the cross-domain ALTO server discovery procedure with parameter X. The result will be the IRD URI of the ALTO server to query.

3.2. Endpoint Cost Service

If an ALTO client wants to query the Endpoint Cost Service (see Section 11.5 of RFC 7285 [RFC7285]) for the costs from source address X to destination address(es) Y (and Z), it has to invoke the cross-domain ALTO server discovery procedure with parameter X. The result will be the IRD URI of the ALTO server to query for the costs from X to Y (and Z)..

3.3. Other ALTO services

TBD. In particular, how to assemble a NxN network map from individual snippets (1xN vectors?) retrieved from different ALTO servers?

4. Implementation, Deployment, and Operational Considerations

4.1. Considerations for ALTO Clients

4.1.1. Resource Consumer Initiated Discovery

To some extent, ALTO requirement AR-32 [RFC6708], i.e., resource consumer initiated ALTO server discovery, can be seen as a special case of cross-domain ALTO server discovery. To that end, an ALTO client embedded in a resouce consumer would have to figure out its own "public" IP address and perform the procedures described in this document on that address. However, due to the widespread deployment of Network Address Translators (NAT), additional protocols and mechanisms such as STUN [RFC5389] would be needed and considerations for UNSAF [RFC3424] apply. Therefore, using the procedures specified in this document for resource consumer based ALTO server discovery is generally NOT RECOMMENDED. Note that a less versatile yet simpler approach for resource consumer initiated ALTO server discovery is specified in [RFC7286].

4.1.2. IPv4/v6 Dual Stack, Multihoming, NAT, and Host Mobility

The algortihm specified in this document can discover ALTO server URIs for a given IP address. The intention is, that a third party (e.g., a resource directory) that receives query messages from a resource consumer can use the source address in these messages to discover suitable ALTO servers for this specific resource consumer.

However, resource consumers (as defined in Section 2 of [RFC5693]) may reside on hosts with more than one IP address, e.g., due to IPv4/v6 dual stack operation and/or multihoming. IP packets sent with different source addresses may be subject to different routing policies and path costs. In some deployment scenarios, it may even be required to ask different sets of ALTO servers for guidance. Furthermore, source addresses in IP packets may be modified en-route by Network Address Translators (NAT).

If a resource consumer queries a resource directory for candidate resource providers, the locally selected (and possibly en-route translated) source address of the query message - as observed by the resource directory - will become the basis for the ALTO server discovery and the subsequent optimization of the resource directory’s reply. If, however, the resource consumer then selects different source addresses to contact returned resource providers, the desired better-than-random "ALTO effect" may not occur.

Therefore, a dual stack or multihomed resource consumer SHOULD either always use the same address for contacting the resource directory and

the resource providers, i.e., overriding the operating system's automatic source IP address selection, or use resource consumer based ALTO server discovery [RFC7286] to discover suitable ALTO servers for every local address and then locally perform ALTO-influenced resource consumer selection and source address selection. Similarly, resource consumers on mobile hosts SHOULD query the resource directory again after a change of IP address, in order to get a list of candidate resource providers that is optimized for the new IP address.

4.2. Deployment Considerations for Network Operators

4.2.1. Separation of Interests

We assume that if two organizations share parts of their DNS infrastructure, i.e., have common in-addr.arpa. and/or ip6.arpa. subdomains, they will also be able to operate a common ALTO server, which still may do redirections if desired or required by policies.

Note that the ALTO server discovery procedure is supposed to produce only a first URI of an ALTO server that can give reasonable guidance to the client. An ALTO server can still return different results based on the client's address (or other identifying properties) or redirect the client to another ALTO server using mechanisms of the ALTO protocol (see Sect. 9 of [RFC7285]).

5. Security Considerations

A high-level discussion of security issues related to ALTO is part of the ALTO problem statement [RFC5693]. A classification of unwanted information disclosure risks, as well as specific security-related requirements can be found in the ALTO requirements document [RFC6708].

The remainder of this section focuses on security threats and protection mechanisms for the cross-domain ALTO server discovery procedure as such. Once the ALTO server's URI has been discovered and the communication between the ALTO client and the ALTO server starts, the security threats and protection mechanisms discussed in the ALTO protocol specification [RFC7285] apply.

5.1. Integrity of the ALTO Server's URI

Scenario Description
: An attacker could compromise the ALTO server discovery procedure or infrastructure in a way that ALTO clients would discover a "wrong" ALTO server URI.

Threat Discussion
: This is probably the most serious security concern related to ALTO server discovery. The discovered "wrong" ALTO server might not be able to give guidance to a given ALTO client at all, or it might give suboptimal or forged information. In the latter case, an attacker could try to use ALTO to affect the traffic distribution in the network or the performance of applications (see also Section 15.1. of [RFC7285]). Furthermore, a hostile ALTO server could threaten user privacy (see also Section 5.2.1, case (5a) in [RFC6708]).

  However, it should also be noted that, if an attacker was able to compromise the DNS infrastructure used for cross-domain ALTO server discovery, (s)he could also launch significantly more serious other attacks (e.g., redirecting various application protocols).

Protection Strategies and Mechanisms
: The cross-domain ALTO server discovery procedure relies on a series of DNS lookups. If an attacker was able to modify or spoof any of the DNS records, the resulting URI could be replaced by a forged URI. The application of DNS security (DNSSEC) [RFC4033] provides a means to limit attacks that rely on modification of the DNS records while in transit. Additional operational precautions for safely operating the DNS infrastructure are required in order to ensure that name servers do not sign forged (or otherwise

"wrong") resource records. Security considerations specific to U-NAPTR are described in more detail in [RFC4848].

A related risk is the impersonation of the ALTO server (i.e., attacks after the correct URI has been discovered). This threat and protection strategies are discussed in Section 15.1 of [RFC7285]. Note that if TLS is used to protect ALTO, the server certificate will contain the host name (CN). Consequently, only the host part of the HTTPS URI will be authenticated, i.e., the result of the ALTO server discovery procedure. The DNS/U-NAPTR based mapping within the cross-domain ALTO server discovery procedure needs to be secured as described above, e.g., by using DNSSEC.

In addition to active protection mechanisms, users and network operators can monitor application performance and network traffic patterns for poor performance or abnormalities. If it turns out that relying on the guidance of a specific ALTO server does not result in better-than-random results, the usage of the ALTO server may be discontinued (see also Section 15.2 of [RFC7285]).

5.2. Availability of the ALTO Server Discovery Procedure

Scenario Description
: An attacker could compromise the cross-domain ALTO server discovery procedure or infrastructure in a way that ALTO clients would not be able to discover any ALTO server.

Threat Discussion
: If no ALTO server can be discovered (although a suitable one exists) applications have to make their decisions without ALTO guidance. As ALTO could be temporarily unavailable for many reasons, applications must be prepared to do so. However, The resulting application performance and traffic distribution will correspond to a deployment scenario without ALTO.

Protection Strategies and Mechanisms
: Operators should follow best current practices to secure their DNS and ALTO (see Section 15.5 of [RFC7285]) servers against Denial-of-Service (DoS) attacks.

5.3. Confidentiality of the ALTO Server's URI

Scenario Description
An unauthorized party could invoke the cross-domain ALTO server discovery procedure, or intercept discovery messages between an authorized ALTO client and the DNS servers, in order to acquire knowledge of the ALTO server URI for a specific IP address.

Threat Discussion
In the ALTO use cases that have been described in the ALTO problem statement [RFC5693] and/or discussed in the ALTO working group, the ALTO server's URI as such has always been considered as public information that does not need protection of confidentiality.

Protection Strategies and Mechanisms
No protection mechanisms for this scenario have been provided, as it has not been identified as a relevant threat. However, if a new use case is identified that requires this kind of protection, the suitability of this ALTO server discovery procedure as well as possible security extensions have to be re-evaluated thoroughly.

5.4. Privacy for ALTO Clients

Scenario Description
An unauthorized party could intercept messages between an ALTO client and the DNS servers, and thereby find out the fact that said ALTO client uses (or at least tries to use) the ALTO service in order to optimize traffic from/to a specific IP address.

Threat Discussion
In the ALTO use cases that have been described in the ALTO problem statement [RFC5693] and/or discussed in the ALTO working group, this scenario has not been identified as a relevant threat.

Protection Strategies and Mechanisms
No protection mechanisms for this scenario have been provided, as it has not been identified as a relevant threat. However, if a new use case is identified that requires this kind of protection, the suitability of this ALTO server discovery procedure as well as possible security extensions have to be re-evaluated thoroughly.

6. IANA Considerations

This document does not require any IANA action.

7. References

7.1. Normative References

[RFC1035] Mockapetris, P., "Domain names - implementation and specification", STD 13, RFC 1035, November 1987.

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

[RFC3596] Thomson, S., Huitema, C., Ksinant, V., and M. Souissi, "DNS Extensions to Support IP Version 6", RFC 3596, October 2003.

[RFC4848] Daigle, L., "Domain-Based Application Service Location Using URIs and the Dynamic Delegation Discovery Service (DDDS)", RFC 4848, April 2007.

7.2. Informative References

[I-D.ietf-alto-deployments] Stiemerling, M., Kiesel, S., Scharf, M., Seidel, H., and S. Previdi, "ALTO Deployment Considerations", draft-ietf-alto-deployments-15 (work in progress), May 2016.

[I-D.kiesel-alto-3pdisc] Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., Tomsu, M., and H. Song, "ALTO Server Discovery Protocol", draft-kiesel-alto-3pdisc-05 (work in progress), March 2011.

[I-D.kiesel-alto-alto4alto] Kiesel, S., "Using ALTO for ALTO server selection", draft-kiesel-alto-alto4alto-00 (work in progress), July 2010.

[I-D.kiesel-alto-ip-based-srv-disc] Kiesel, S. and R. Penno, "Application-Layer Traffic Optimization (ALTO) Anycast Address", draft-kiesel-alto-ip-based-srv-disc-03 (work in progress), July 2014.

[I-D.kist-alto-3pdisc] Kiesel, S., Krause, K., and M. Stiemerling, "Third-Party ALTO Server Discovery (3pdisc)", draft-kist-alto-3pdisc-05 (work in progress), January 2014.

[RFC3424] Daigle, L. and IAB, "IAB Considerations for UNilateral Self-Address Fixing (UNSAF) Across Network Address Translation", RFC 3424, November 2002.

[RFC4033] Arends, R., Austein, R., Larson, M., Massey, D., and S. Rose, "DNS Security Introduction and Requirements", RFC 4033, March 2005.

[RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing, "Session Traversal Utilities for NAT (STUN)", RFC 5389, October 2008.

[RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, October 2009.

[RFC6708] Kiesel, S., Previdi, S., Stiemerling, M., Woundy, R., and Y. Yang, "Application-Layer Traffic Optimization (ALTO) Requirements", RFC 6708, September 2012.

[RFC7216] Thomson, M. and R. Bellis, "Location Information Server (LIS) Discovery Using IP Addresses and Reverse DNS", RFC 7216, April 2014.

[RFC7285] Alimi, R., Penno, R., Yang, Y., Kiesel, S., Previdi, S., Roome, W., Shalunov, S., and R. Woundy, "Application-Layer Traffic Optimization (ALTO) Protocol", RFC 7285, September 2014.

[RFC7286] Kiesel, S., Stiemerling, M., Schwan, N., Scharf, M., and H. Song, "Application-Layer Traffic Optimization (ALTO) Server Discovery", RFC 7286, June 2014.

Appendix A. Requirements for ALTO Cross-Domain Server Discovery

A solution for the problem described in the previous section would be an ALTO Cross-Domain Server Discovery system. This section itemizes requirements.

A.1. Discovery Client Application Programming Interface

The discovery client will be called through some kind of application programming interface (API) and the parameters will be an IP address and, for purposes of extensibility, a service identifier such as "ALTO". It will return one or more URI(s) that offers the requested service ("ALTO") for the given IP address.

In other words, the client would be used to retrieve a mapping:

(IP address, "ALTO") -> IRD-URI(s)

where IRD-URI(s) is one or more URI(s) of Information Resource Directories (IRD, see Section 9 of [RFC7285]) of ALTO server(s) that can give reasonable guidance to a resource consumer with the indicated IP address.

A.2. Data Storage and Authority Requirements

The information for mapping IP addresses and service parameters to URIs should be stored in a - preferably distributed - database. It must be possible to delegate administration of parts of this database. Usually, the mapping from a specific IP address to an URI is defined by the authority that has administrative control over this IP address, e.g., the ISP in residential access networks or the IT department in enterprise, university, or similar networks.

A.3. Cross-Domain Operations Requirements

The cross-domain server discovery mechanism should be designed in such a way that it works across the public Internet and also in other IP-based networks. This in turn means that such mechanisms cannot rely on protocols that are not widely deployed across the Internet or protocols that require special handling within participating networks. An example is multicast, which is not generally available across the Internet.

The ALTO Cross-Domain Server Discovery protocol must support gradual deployment without a network-wide flag day. If the mechanism needs some kind of well-known "rendezvous point", re-using an existing infrastructure (such as the DNS root servers or the WHOIS database) should be preferred over establishing a new one.

A.4. Protocol Requirements

The protocol must be able to operate across middleboxes, especially across NATs and firewalls.

The protocol shall not require any pre-knowledge from the client other than any information that is known to a regular IP host on the Internet.

A.5. Further Requirements

The ALTO cross domain server discovery cannot assume that the server discovery client and the server discovery responding entity are under the same administrative control.

Appendix B. ALTO and Tracker-based Peer-to-Peer Applications

The ALTO protocol specification [RFC7285] details how an ALTO client can query an ALTO server for guiding information and receive the corresponding replies. However, in the considered scenario of a tracker-based P2P application, there are two fundamentally different possibilities where to place the ALTO client:

1. ALTO client in the resource consumer ("peer")

2. ALTO client in the resource directory ("tracker")

In the following, both scenarios are compared in order to explain the need for ALTO queries on behalf of remote resource consumers.

In the first scenario (see Figure 2), the resource consumer queries the resource directory for the desired resource (F1). The resource directory returns a list of potential resource providers without considering ALTO (F2). It is then the duty of the resource consumer to invoke ALTO (F3/F4), in order to solicit guidance regarding this list.

In the second scenario (see Figure 4), the resource directory has an embedded ALTO client. After receiving a query for a given resource (F1) the resource directory invokes this ALTO client to evaluate all resource providers it knows (F2/F3). Then it returns a, possibly shortened, list containing the "best" resource providers to the resource consumer (F4).

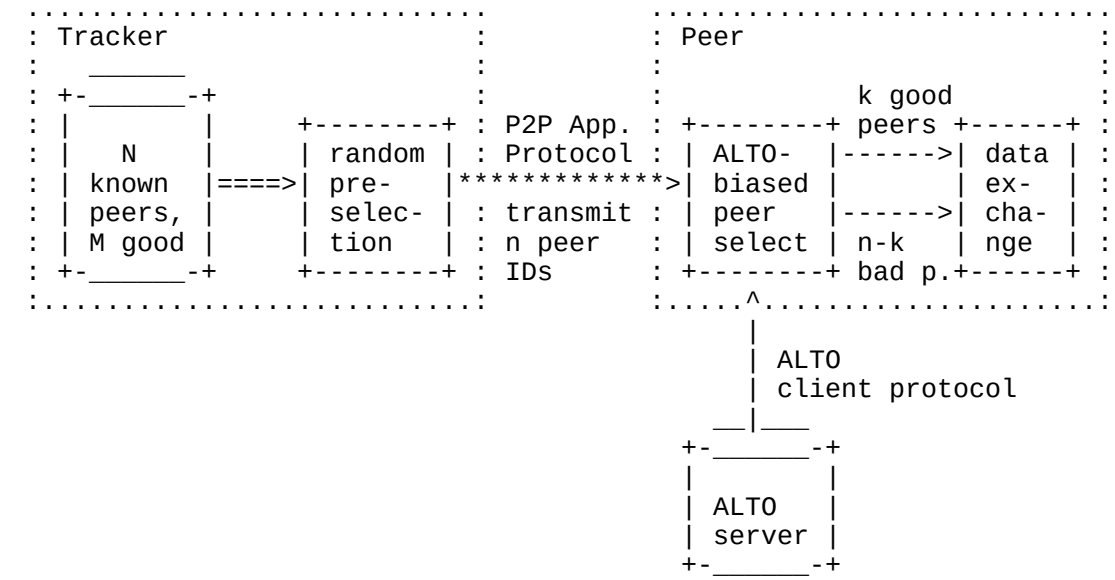
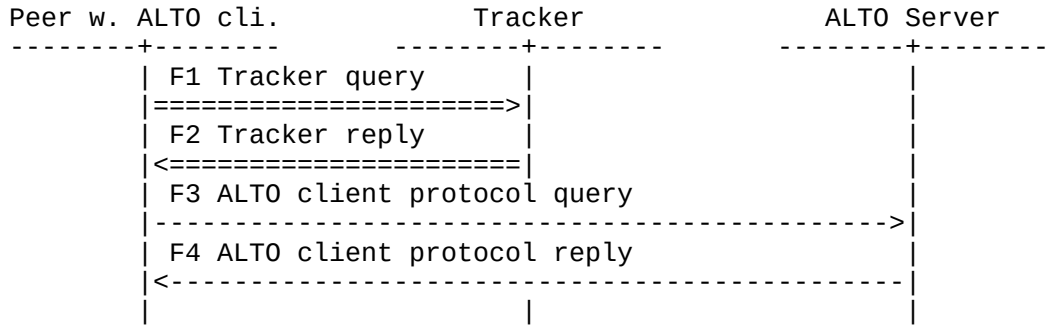


Figure 1: Tracker-based P2P Application with random peer preselection



==== Application protocol (i.e., tracker-based P2P app protocol)
---- ALTO client protocol

Figure 2: Basic message sequence chart for resource consumer-initiated ALTO query

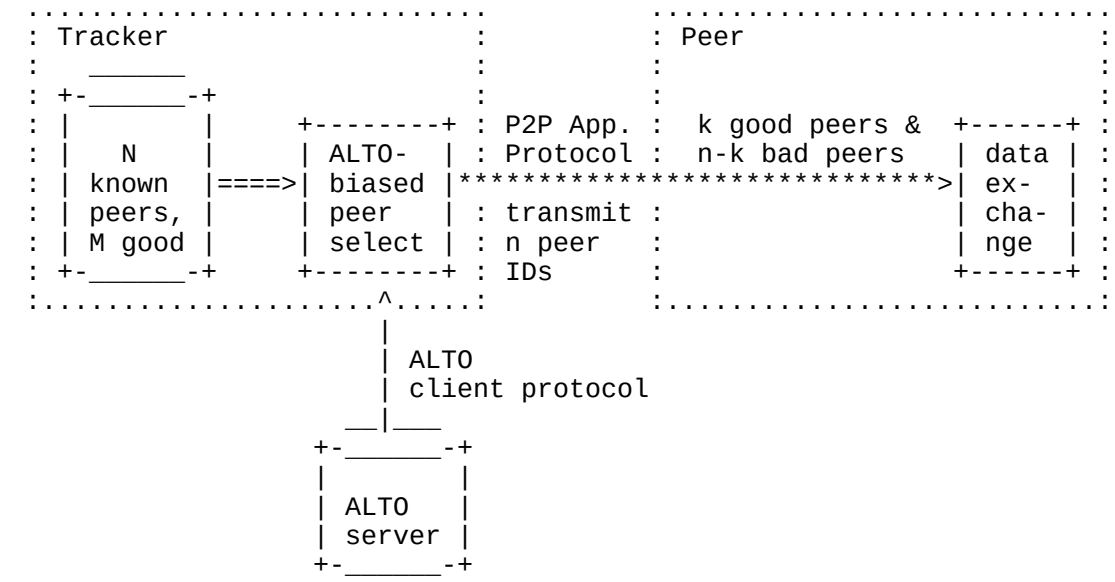
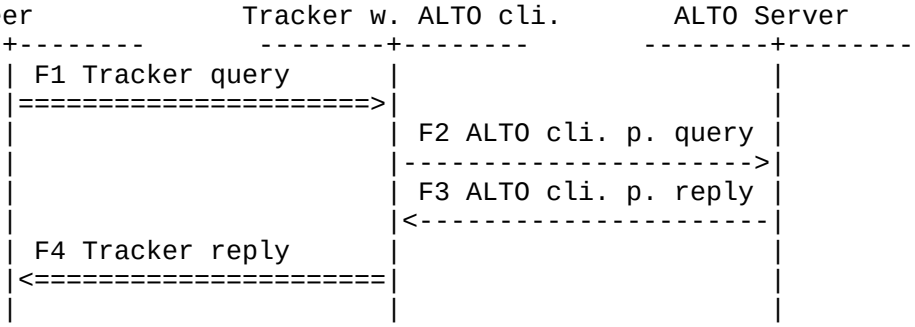


Figure 3: Tracker-based P2P Application with ALTO client in tracker



==== Application protocol (i.e., tracker-based P2P app protocol)
---- ALTO client protocol

Figure 4: Basic message sequence chart for ALTO query on behalf of remote resource consumer

Note: the message sequences depicted in Figure 2 and Figure 4 may occur both in the target-aware and the target-independent query mode (c.f. [RFC6708]). In the target-independent query mode no message exchange with the ALTO server might be needed after the tracker query, because the candidate resource providers could be evaluated using a locally cached "map", which has been retrieved from the ALTO

server some time ago.

The problem with the first approach is, that while the resource directory might know thousands of peers taking part in a swarm, the list returned to the resource consumer is usually shortened for efficiency reasons. Therefore, the "best" (in the sense of ALTO) potential resource providers might not be contained in that list anymore, even before ALTO can consider them.

For illustration, consider a simple model of a swarm, in which all peers fall into one of only two categories: assume that there are "good" ("good" in the sense of ALTO's better-than-random peer selection, based on an arbitrary desired rating criterion) and "bad' peers only. Having more different categories makes the maths more complex but does not change anything to the basic outcome of this analysis. Assume that the swarm has a total number of N peers, out of which are M "good" and N-M "bad" peers, which are all known to the tracker. A new peer wants to join the swarm and therefore asks the tracker for a list of peers.

If, according to the first approach, the tracker randomly picks n peers from the N known peers, the result can be described with the hypergeometric distribution. The probability that the tracker reply contains exactly k "good" peers (and n-k "bad" peers) is:

$$P(X=k) = \frac{\binom{m}{k}\binom{N-m}{n-k}}{\binom{N}{n}}$$

with $\binom{n}{k} = \frac{n!}{k!\,(n-k)!}$ and $n! = n * (n-1) * (n-2) * .. * 1$

The probability that the reply contains at most k "good" peers is: P(X<=k)=P(X=0)+P(X=1)+..+P(X=k).

For example, consider a swarm with N=10,000 peers known to the tracker, out of which M=100 are "good" peers. If the tracker randomly selects n=100 peers, the formula yields for the reply: P(X=0)=36%, P(X<=4)=99%. That is, with a probability of approx. 36% this list does not contain a single "good" peer, and with 99% probability there are only four or less of the "good" peers on the

list. Processing this list with the guiding ALTO information will ensure that the few favorable peers are ranked to the top of the list; however, the benefit is rather limited as the number of favorable peers in the list is just too small.

Much better traffic optimization could be achieved if the tracker would evaluate all known peers using ALTO, and return a list of 100 peers afterwards. This list would then include a significantly higher fraction of "good" peers. (Note, that if the tracker returned "good" peers only, there might be a risk that the swarm might disconnect and split into several disjunct partitions. However, finding the right mix of ALTO-biased and random peer selection is out of the scope of this document.)

Therefore, from an overall optimization perspective, the second scenario with the ALTO client embedded in the resource directory is advantageous, because it is ensured that the addresses of the "best" resource providers are actually delivered to the resource consumer. An architectural implication of this insight is that the ALTO server discovery procedures must support ALTO queries on behalf of remote resource consumers. That is, as the tracker issues ALTO queries on behalf of the peer which contacted the tracker, the tracker must be able to discover an ALTO server that can give guidance suitable for that respective peer.

Appendix C. Contributors List and Acknowledgments

The initial version of this document was co-authored by Marco Tomsu (Alcatel-Lucent).

This document borrows some text from [RFC7286], as it was historically part of that memo. Special thanks to Michael Scharf and Nico Schwan.

Authors' Addresses

Sebastian Kiesel
University of Stuttgart Information Center
Allmandring 30
Stuttgart 70550
Germany

Email: ietf-alto@skiesel.de
URI: http://www.rus.uni-stuttgart.de/nks/

Martin Stiemerling
University of Applied Sciences Darmstadt, Computer Science Dept.
Haardtring 100
Darmstadt 64295
Germany

Phone: +49 6151 16 7938
Email: mls.ietf@gmail.com
URI: http://ietf.stiemerling.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

S. Randriamasy
Nokia Bell Labs
R. Yang
Yale University
Q. Wu
Huawei
L. Deng
China Mobile
N. Schwan
Thales Deutschland

July 08, 2016

ALTO Cost Calendar
draft-randriamasy-alto-cost-calendar-06

Abstract

The goal of Application-Layer Traffic Optimization (ALTO) is to bridge the gap between network and applications by provisioning network related information in order to allow applications to make network informed decisions. The present draft extends the ALTO cost information so as to broaden the decision possibilities of applications to not only decide 'where' to connect to, but also 'when'. This is useful to applications that need to schedule their data transfers and connections and have a degree of freedom to do so. ALTO guidance to schedule application traffic can also efficiently help for load balancing and resources efficiency. Besides, the ALTO Cost Calendar allows to schedule the ALTO requests themselves and thus to save a number of ALTO transactions.

This draft proposes new capabilities and attributes on filtered cost maps and endpoint costs enabling an ALTO Server to provide "Cost Calendars". These capabilities are applicable to time-sensitive ALTO metrics. With ALTO Cost Calendars, an ALTO Server exposes ALTO Cost Values in JSON arrays where each value corresponds to a given time interval. The time intervals as well as other Calendar attributes are specified in the IRD and ALTO Server responses.

Requirements Language

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

IETF is currently standardizing the ALTO protocol which aims at providing guidance to overlay applications needing to select one or several hosts from a set of candidates able to provide a desired resource. This guidance is based on parameters that affect performance and efficiency of the data transmission between the hosts such as the topological distance. The goal of ALTO is to improve the Quality of Experience (QoE) in the application while optimizing resource usage in the underlying network infrastructure.

The ALTO protocol in [RFC7285] specifies a Network Map which defines groupings of endpoints in provider-defined network regions (called PIDs). The Cost Map Service, Endpoint Cost Service (ECS) and Endpoint Ranking Service then provide ISP-defined costs and rankings for connections among the specified endpoints and PIDs and thus incentives for application clients to connect to ISP preferred locations, e.g. to reduce their costs. ALTO intentionally avoids provisioning realtime information as explained in the ALTO Problem Statement [RFC5693] and ALTO Requirements [RFC5693].Thus the current

Cost Map and Endpoint Cost Service are providing, for a given Cost Type, exactly one path cost value. Applications have to query one of these two services to retrieve the currently valid cost values. They therefore need to plan their ALTO information requests according to their own estimation of the frequency of cost value change.

With [RFC7285], an ALTO client should interpret the returned costs as those at the query moment. However, Network costs can fluctuate, e.g. due to diurnal patterns of traffic demand or planned events such as network maintenance, holidays or highly publicized events. Providing network costs for only the current time thus may not be sufficient, in particular for applications that can schedule their traffic in a span of time, for example by deferring backup to night during traffic trough.

In case the ALTO Cost value changes are predicable over a certain period of time and the application does not require immediate data transfer, it can save time to get the whole set of cost values over this period in one ALTO response. Using them to schedule data transfers would allow to optimise the network resources usage and QoE. ALTO Clients and Servers can also minimize their workload by an appropriate scheduling of their data exchanges.

This document extends RFC7285 to allow an ALTO server to provide network costs for a given duration of time. A sequence of network costs across a time span for a given pair of network locations is named an "ALTO Cost Calendar". The Filtered Cost Map Service and Endpoint Cost Service are extended to provide Cost Calendars. In addition to this functional ALTO enhancement, we expect to further gain on storage and on the wire data exchange by gathering multiple Cost Values for one Cost Type into one single ALTO Server response.

In this draft an "ALTO Cost Calendar" is specified by information resources capabilities that are applicable to time-sensitive ALTO metrics. An ALTO Cost Calendar exposes ALTO Cost Values in JSON arrays where each value corresponds to a given time interval. The time intervals as well as other Calendar attributes are specified in the IRD and in the Server response to allow the ALTO Client to interpret the received ALTO values. Last, the proposed extensions for ALTO Calendars are applicable to any Cost Mode and they ensure backwards compatibility with legacy ALTO clients.

In the rest of this document, Section 2 provides the design characteristics. Sections 3 and 4 define the formal specification for the IRD and the information resources. Section 5 provides non-normative use cases to illustrate the usage of cost calendars. IANA considerations and security considerations will be completed in further versions.

2. Overview of ALTO Cost Calendars

An ALTO Cost calendar provided by the ALTO Server provides 2 information items:

- an array of values for a given metric, where each value corresponds to a time interval, wher the value array can sometimes be a cyclic pattern that repeats a certain number of times.

- attributes describing the time scope of the calendar, allowing an ALTO Client to properly interpret the values, such as the size and number of the intervals and the date of the starting point of the calendar.

An ALTO Cost Calendar can be used like a "time table" to figure out the best time to schedule data transfers and also to proactively manage appllication traffic given predictable events such as flash crowds, traffic intensive holidays and network maintenance. An ALTO Cost Calendar may be viewed as a synthetic abstraction of real measurements that can be historic or be a prediction for upcoming time periods.

Most likely, the ALTO Cost Calendar would be used for the Endpoint Cost Service, assuming that a limited set of feasible Endpoints for a non-real time application is already identified, that they do not need to be accessed immediately and that their access can be scheduled within a given time period. The Filtered Cost Map service is also applicable as long as the size of the Map allows it.

2.1. ALTO Cost Calendar information features

The Calendar attributes are provided in the IRD and in ALTO Server responses. The IRD announces attributes with dateless values in its information resources capabilities, where as attributes with time dependent values are provided in the "meta" of Server responses. The ALTO Cost Calendar attributes provide the following information:

- attributes to interpret the time scope of the Calendar value array:

  * generic time zone,

  * applicable time interval for each calendar value: combining numbers and time units to reflect for example: 1 hour, 2 minutes, 10 seconds, 1 week, 1 month,

  * duration of the Calendar: e.g. the number of intervals provided in the calendar.

- "calendar-start-date": specifying when the calendar starts, that is to which date the first value of the cost calendar is applicable.

- "repeated": an optional attribute indicating how many iterations the provided calendar will have the same values, that the server may use to allow the client to schedule its next request and thus save its own workload by avoiding to process useless requests.

2.2. ALTO Calendar design characteristics

This draft introduces new capabilities and attributes that specify an ALTO Cost Calendar. The protocol extension placeholders are: the IRD, the ALTO requests and responses for Cost calendars.

Extensions are designed to be light and ensure backwards compatibility with base protocol ALTO Clients and with other extensions. It uses section 8.3.7 "Parsing of Unknown Fields" of RFC7285 that writes: "Extensions may include additional fields within JSON objects defined in this document. ALTO implementations MUST ignore unknown fields when processing ALTO messages."

The calendar specific capabilities are integrated in the information resources of the IRD and in the "meta" member of ALTO responses to Cost Calendars requests. A calendar and its capabilities are associated to a given information resource and within this information resource to a given cost type. This design has several advantages:

- it does not introduce a new mode,

- it does not introduce new media types,

- it allows an ALTO Server to offer calendar capabilities on a cost type, with attributes values adapted to each information resource.

The Applicable Calendared information resources are:

- the Filtered Cost Map,

- the Endpoint Cost Map.

The ALTO Server can choose in which frequency it provides cost Calendars to ALTO Clients. It may either provide calendar updates starting at the request date, or carefully schedule its updates so as to take profit from a potential repetition/periodicity of calendar values.

2.2.1. ALTO Cost Calendar for all cost modes

Calendars are well-suited for values encoded in the ’numerical’ mode. However, Calendars can also represent any metric considered as time-sensitive by an ALTO Server. For example, types of Cost values such as JSONBool can also be expressed as calendars, as states may be "true" or "false" depending on given time periods or likewise, values represented by strings, such as "medium", "high", "low", "blue", "open" .

Note also that a Calendar is applicable as well to time-sensitive metrics provided in the ’ordinal’ mode, if these values are time-sentitive and their update is carefully managed by the ALTO Server.

2.2.2. Compatibility with legacy ALTO Clients

The ALTO protocol extensions for Cost Calendars have been defined so as to ensure that Calendar capable ALTO Servers can provide legacy ALTO Clients with legacy information resources as well. That is a legacy ALTO Client can request resources and receive responses as specified in RFC7285.

For compatibility with legacy ALTO Clients specified in RFC7285, calendared information resources are not applicable for Cost Maps for the following reason: a legacy ALTO client would receive a Calendared Cost Map via an HTTP ’GET’ command. As specified in section 8.3.7 of RFC7285, it will ignore the Calendar Attributes indicated in the "meta" of the responses. Therefore, lacking information on calendar attributes, it will not be able to correctly interpret and process the values of the received array of calendar cost values.

3. ALTO Calendar specification: IRD extensions

The Calendar attributes in the IRD information resources capabilities carry constant dateless values. A calendars is associated to an information resource rather than a cost type. For example, a Server can provide a "routingcost" values calendar for the Filtered Cost Map Service at a granularity of one day and a "routingcost" values calendar for the Endpoint Cost service at a finer granularity but for a limited number of endpoints.

NOTE : to cope with existing representation fomats and proposed unified ALTO naming schemes proposed in the WG, the names given in the current proposal may be revised in further versions.

3.1. Calendar attributes in the IRD resources capabilities

When for an applicable resource , an ALTO Server provides a Cost Calendar for a given Cost Type, it MUST indicate this in the IRD capabilities of this resource, by an object of type 'CalendarAttributes', associated to this Cost Type and specified below.

The capabilities of a Calendar aware information resource entry have a member named "calendar-attributes" which is an array of objects of type CalendarAttributes. The array has as many values as cost-type-names announced for the resource. It is necessary to use an array because of resources such as Filtered Cost Map and Endpoint Cost Map, for which the member "cost-type-names" is an array of 1 or more values. If for a given cost-type-name of this ressource no Calendar attributes are defined, the ALTO Server MUST replace that value in the array by the symbol 'null'.

RULE: a member "calendar-attributes" MUST appear only once for each applicable cost type name of a resource entry. If "calendar-attributes" are specified several times for a same "cost-type-name" in the capabilities of a resource entry, the ALTO client SHOULD ignore any calendar capabilities on this "cost-type-name" for this entry.

CalendarAttributes calendar-attributes <1..*>;

```
object{
  [JSONString   cost-type-name;]
  JSONString    time-interval-size;
  JSONNumber    number-of-intervals;
} CalendarAttributes;
```

- "cost-type-name":
  - an optional member indicating the cost-type-name in the IRD entry to which the capabilities apply. If this not present, it MUST be assumed to correspond to its index in the "cost-type-names" list of the IRD resource entry.
- "time-interval-size":
  - is the duration of an ALTO calendar time interval, expressed as a time unit appended to the number of these units. The time unit, ranges from "second" to "year". The number is encoded with an integer. Example values are: "5 minute" , "2 hour", meaning that each calendar value applies on a time interval that lasts respectively 5 minutes and 2 hours.

- "number-of-intervals":
  - the integer number of values of the cost calendar array, at least equal to 1.

- Attribute "cost-type-name" , if used, provides a better readability to the calendar attributes specified in the IRD and avoids confusion with calendar attributes of other cost-types.

- Multiplying Attributes ’time-interval-size’ and ’number-of-intervals’ provides the duration of the provided calendar. For example an ALTO Server may provide a calendar for ALTO values changing every ’time-interval-size’ equal to 5 minutes. If ’number-of-intervals’ has the value 12, then the duration of the provided calendar is "1 hour".

3.2. Calendars in a delegate IRD

One option to clarify IRD resources is that a "root" ALTO Server implementing base protocol resources delegates "specialized" information resources such as the ones providing Cost Calendars to another ALTO Server running in a subdomain specified with its URI in the "root" ALTO Server. This option is described in Section 9.2.4 "Delegation using IRDs" of RFC7285.

This document provides an example, where a "root" ALTO Server runs in a domain called "alto.example.com". It delegates the announcement of Calendars capabilities to an ALTO Server running in a subdomain called "custom.alto.example.com". The location of the "delegate Calendar IRD" is assumed to be indicated in the "root" IRD by the resource entry: "custom-calendared-resources".

Another advantage is that some Cost Types for some resources may be more advantageous as Cost Calendars and it makes few sense to get them as a single value. For example, Cost Types with predictable and frequently changing values, calendared in short time intervals such as a minute.

3.3. Example IRD with ALTO Cost Calendars

The cost types in this example are either specified in the base ALTO protocol or may be proposed in other drafts see [draft-wu-alto-te-metrics]. In this example the available cost metrics are indicated in the "meta" field by cost type names "num-routingcost", "num-latency", "num-pathbandwidth" and "string-quality-status". Metrics "routingcost" , ’latency’ and ’Availbandwidth’ are available in the "numerical" Cost Mode. Metric "quality-status" is available in the "string" Cost Mode.

This ALTO server does not provide a calendar for cost type name num-AShopcount.

The example IRD includes 2 particular URIs providing calendars:

- "http://custom.alto.example.com/calendar/costmap/filtered": a filtered cost map in which calendar capabilities are indicated for cost type names: "num-routingcost", "num-pathbandwidth" and "string-service-status",

- "http://custom.alto.example.com/endpointcost/calendar/lookup": an endpoint cost map in which in which calendar capabilities are indicated for cost type names: "num-routingcost", "num-TEpktloss", "num-pathbandwidth", "string-service-status".

The design of the Calendar capabilities allows that some calendars on a cost type name are available in several information resources with different Calendar Attributes. This is the case for calendars on "num-routingcost", "num-pathbandwidth" and "string-service-status" , available in both the Filtered Cost map and Endpoint Cost map service, but with different time interval sizes for "num-pathbandwidth" and "string-service-status".

```
GET /calendars-directory HTTP/1.1
Host: custom.alto.example.com
Accept: application/alto-directory+json,application/alto-error+json
---------------

HTTP/1.1 200 OK
Content-Length: [TODO]
Content-Type: application/alto-directory+json

   {
     "meta" : {
        "cost-types": {
           "num-routingcost": {
              "cost-mode" : "numerical",
              "cost-metric" : "routingcost"
           },
           "num-latency": {
              "cost-mode"  : "numerical",
              "cost-metric": "latency"
           },
           "num-pathbandwidth": {
              "cost-mode"  : "numerical",
              "cost-metric": "Availbandwidth",
           },
           "string-qual-status": {
```

```
              "cost-mode"  : "string",
              "cost-metric": "quality-status",
           }
         ... other meta ...
       },

    "resources" : {
       "filtered-cost-map-calendar" : {
           "uri" : "http://custom.alto.example.com/calendar/costmap/filtered",
           "media-type" : "application/alto-costmap+json",
           "accepts" : "application/alto-costmapfilter+json",
           "capabilities" : {
              "cost-constraints" : true,
              "cost-type-names"  : [ "num-routingcost", "num-pathbandwidth",
                                     "string-service-status" ],
              "calendar-attributes" : [
                 {"cost-type-names" : [ "num-routingcost", "num-pathbandwidth" ]
,
                  "time-interval-size" : "1 hour",
                  "number-of-intervals" : 24
                 },
                 {"cost-type-names" : "string-service-status",
                  "time-interval-size" : "30 minute",
                  "number-of-intervals" : 48
                 }
               ] // end calendar-attributes
              "uses": [ "my-default-network-map" ]
              }
           },

       "endpoint-cost-calendar-map" : {
         "uri" : "http://custom.alto.example.com/calendar/endpointcost/calendar/
lookup",
         "media-types" : [ "application/alto-endpointcost+json" ],
         "accepts" : [ "application/alto-endpointcostparams+json" ],
         "capabilities" : {
           "cost-constraints" : true,
           "cost-type-names" : [ "num-routingcost", "num-latency",
                                 "num-pathbandwidth", "string-service-status" ],
           "calendar-attributes" : [
              {"cost-type-names" : "num-routingcost",
               "time-interval-size" : "1 hour",
               "number-of-intervals" : 24
              },
              {"cost-type-names" : "latency",
               "time-interval-size" : "5 minute",
               "number-of-intervals" : 12
              },
              {"cost-type-names" : "num-pathbandwidth",
               "time-interval-size" : "1 minute",
```

```
              "number-of-intervals" : 60
             },
             {"cost-type-names" : "string-service-status",
              "time-interval-size" : "2 minute",
              "number-of-intervals" : 30
             },
            ]
          "uses": [ "my-default-network-map" ]
       } // ECM capab
    } //info resource N
} // ressources
```

In this example IRD, for the filtered cost map service, all calendars have a duration of 1 day and start in the "request-date" mode, that is the "date" of first value of the array belongs to the time interval "containing" the date of the request.

- the Calendar for ’num-routingcost’: is an array of 24 values each provided on a time interval lasting 1 hour.
- the Calendar for ’num-pathbandwidth’: is an array of 24 values each provided on a time interval lasting 1 hour.
- the Calendar for "string-service-status": "is an array of 48 values each provided on a time interval lasting 30 minutes.

For the endpoint cost map service, the cost calendars have a duration of 1 day for "num-routingcost" and 1 hour for the 3 other cost type names.

- the Calendar for ’num-routingcost’: is an array of 24 values each provided on a time interval lasting 1 hour.
- the Calendar for ’latency’’: is an array of 12 values each provided on a time interval lasting 5 minutes.
- the Calendar for ’num-pathbandwidth’: is an array of 60 values each provided on a time interval lasting 1 minute.
- the Calendar for "string-service-status": "is an array of 30 values each provided on a time interval lasting 2 minutes.

4. ALTO Calendar specification: Service Information Resources

This section documents the individual information resources defined to provide the Calendared information services defined in this document.

The reference time zone for the provided time values is GMT because the option chosen to express the time format is the HTTP header fields format:

Date: Tue, 15 Nov 1994 08:12:31 GMT

4.1. Calendar extensions for Filtered Cost Maps

A legacy ALTO client requests and gets filtered cost map responses as specified in RFC7285.

4.1.1. Calendar extensions in Filtered cost map requests

The input parameters of a "legacy" request for a filtered cost map, defined by object ReqFilteredCostMap in section 11.3.2 of RFC7285, are augmented with one additional member.

A Calendar-aware ALTO client requesting a Calendar on a given Cost Type for a Filtered Cost Map resource having Calendar capabilities MUST add the following field to its input parameters:

JSONBoolean calendared<1..*>;

This field is an array of 1 to N boolean values, where N is the number of requested metrics. Each boolean value indicates whether or not the ALTO Server should provide the values for this Cost Type as a calendar.

This field MUST NOT be specified if member "calendar-attributes" is not present or has the value ’false’ for this information resource.

A Calendar-aware ALTO client supporting single cost type values, as specified in RFC7285, MUST provide an array of 1 element: "calendared" : [true],

A Calendar-aware ALTO client that is also Multi-Cost aware MUST provide an array of N values set to "true" or "false", depending whether it wants the applicable Cost Type values as a single or calendared value.

If this field is not present, it MUST be assumed to have only values equal to "false".

4.1.2. Calendar extensions in Filtered Cost map responses

The calendared costs are JSONArrays instead of JSONNumbers for the legacy ALTO implementation. All arrays have a number of values equal to 'number-of-intervals'.

The "meta" field of a Calendared Filtered Cost map response MUST include at least:

- if the ALTO Client supports cost values for one Cost Type at a time only: the "meta" fields specified specified in RFC 7285 for these information service responses:
  - "dependent-vtags ",
  - "cost-type" field.
- if the ALTO Client supports cost values for several Cost Types at a time, as specified in [draft-ietf-alto-multi-cost] : the "meta" fields specified in [draft-ietf-alto-multi-cost] for these information service responses:
  - "dependent-vtags ",
  - "multi-cost-types" field.

In addition, the "meta" field of a Calendared Filtered Cost map response MUST include the member "calendar-response-attributes" for the requested information resource, together with the values provided by the ALTO Server for these attributes. This member is an array of objects of type "CalendarResponseAttributes", defined as follows:

```
CalendarResponseAttributes calendar-response-attributes <1..*>;

object{
  JSONString    calendar-start-time;
  JSONString    time-interval-size;
  JSONNumber    number-of-intervals;
  [JSONNumber   repeated;]            [OPTIONAL]
} CalendarResponseAttributes;
```

- "calendar-start-time": indicates the date at which the first value of the calendar applies. By default, the value provided for the "calendar-start-time" attribute SHOULD be no later than the request date.
- "time-interval-size": as specified in section "Calendar attributes in the IRD resources capabilities",

- "number-of-intervals": as specified in section "Calendar attributes in the IRD resources capabilities",

- "repeated": is an optional field provided for Calendars. It is an integer N greater or equal to '1' that indicates how many iterations of the calendar value array starting at the date indicated by "calendar-start-time" have the same values. The number N includes the provided iteration.

Using the member "repeated" helps minimizing on the wire data exchange: by providing it, an ALTO Server will avoid unecessary processing of requests for Calendars with unchanged values while it allows ALTO Clients to save their resources as well.

For example: if the "calendar-start-time" member has value "Mon, 30 Jun 2014 at 00:00:00 GMT" and if the value of member "repeated" is equal to 4, it means that the calendar values are the same values on Monday, Tuesday, Wednesday and Thursday. The ALTO Client thus may use the same calendar for the next 4 duration periods following "calendar-start-time".

4.1.3. Example transaction for a FCM with a bandwidth Calendar

An example of non-real time information that can be provisioned in a 'calendar' is the expected path bandwidth. While the transmission rate can be measured in real time by end systems, the operator of a data center is in the position of formulating preferences for given paths, at given time periods for example to avoid traffic peaks due to diurnal usage patterns. In this example, we assume that an ALTO Client requests a bandwidth calendar as specified in the IRD to shedule its bulk data transfers as described in the use cases.

In the example IRD, calendars for cost type name "num-pathbandwidth" are available for the information resources: "filtered-cost-calendar-map" and "endpoint-cost-calendar-map". The ALTO Client requests a calendar for "num-pathbandwidth" via a POST request for a filtered cost map.

We suppose in this example that the ALTO Client sends its request on Tuesday July 1st 2014 at 13:15

```
  POST /calendar/costmap/filtered HTTP/1.1
  Host: alto.example.com
  Content-Length: [TODO]
  Content-Type: application/alto-costmapfilter+json
  Accept: application/alto-costmap+json,application/alto-error+json

  {
    "cost-type" : {"cost-mode" : "numerical", "cost-metric" : "Availbandwidth"},
    "calendared" : [true],

    "pids" : {
       "srcs" : [ "PID1", "PID2" ],
       "dsts" : [ "PID1", "PID2", "PID3" ]
    }
  }


  HTTP/1.1 200 OK
  Content-Length: [TODO]
  Content-Type: application/alto-costmap+json

  {
    "meta" : {
      "dependent-vtags" : [...],
      "cost-type" : {"cost-mode" : "numerical", "cost-metric" : "Availbandwidth"
},
      "calendar-response-attributes" : [
         "calendar-start-time" : Tue, 1 Jul 2014 13:00:00 GMT,
         "time-interval-size" : "2 hour",
         "numb-intervals" : 12
       ]
     },

    "cost-map" : {
       "PID1": { "PID1": [v1,v2, ... v12],
                 "PID2": [v1,v2, ... v12],
                 "PID3": [v1,v2, ... v12] },
       "PID2": { "PID1": [v1,v2, ... v12],
                 "PID2": [v1,v2, ... v12],
                 "PID3": [v1,v2, ... v12] }
    }
  }
```

4.2. Calendar extensions in the Endpoint Cost Map Service

This document extends the Endpoint Cost Service, as defined in {11.5.1} of [RFC7285], by adding new input parameters and capabilities, and by returning JSONArrays instead of JSONNumbers as the cost values. The media type (11.5.1.1} and HTTP method (11.5.1.2} are unchanged.

4.2.1. Calendar specific input in Endpoint cost map requests

The extensions to the requests for calendared Endpoint Cost Maps are the same as for the Filtered Cost Map Service, specified in section XXXX of this draft.

The ReqEndpointCostMap object for a Calendared ECM request will have the following format:

```
object {
  CostType       cost-type;
  [JSONBoolean    calendared<1..*>;]
  EndpointFilter endpoints;
} ReqEndpointCostMap;

object {
  [TypedEndpointAddr srcs<0..*>;]
  [TypedEndpointAddr dsts<0..*>;]
} EndpointFilter;
```

4.2.2. Calendar attributes in the Endpoint Cost Map response

The "meta" field of a Calendared Endpoint Cost map response MUST include at least:

- if the ALTO Client supports cost values for one Cost Type at a time only: the "meta" fields specified in {11.5.1.6} of RFC 7285 for the Endpoint Cost response:
  - "cost-type" field.
- if the ALTO Client supports cost values for several Cost Types at a time, as specified in [draft-ietf-alto-multi-cost] : the "meta" fields specified in [draft-ietf-alto-multi-cost] for the the Endpoint Cost response:
  - "multi-cost-types" field.

If the client request does not provide member "calendared" or if it provides it with a value equal to 'false', then the ALTO Server response is exactly as specified in the above cited references.

If the ALTO client provides member "calendared" with a value equal to 'true' in the input parameters, the "meta" member of a Calendared Endpoint Cost Map response MUST include the same addifional member "calendar-response-attributes" as specified for the Filtered Cost Map Service. The Server response is thus changed as follows:

- the "meta" member has one additional field "CalendarResponseAttributes", as specified for the Filtered Cost Map Service,

- the calendared costs are JSONArrays instead of JSONNumbers for the legacy ALTO implementation. All arrays have a number of values equal to 'number-of-intervals'.

4.2.3. Example transaction for the ECS with a routingcost Calendar

Let us assume an Application Client is located in an end sytem with limited resources and having an access to the network that is either intermittent or provides an acceptable quality in limited but predictable time periods. Therefore, it needs to both schedule its resources greedy networking activities and its ALTO transactions.

The Application Client has the choice to trade content or resources with a set of Endpoints and needs to decide with which one it will connect and at what time. For instance, the Endpoints are spread in different time-zones, or have intermittent access. In this example, the 'routingcost' is assumed to be time sentitive with values provided as ALTO Calendars.

The ALTO Client associated to the Application Client queries an ALTO Calendar on 'routingcost' and will get the Calendar covering the 24 hours time period "containing" the date and time of the ALTO client request.

For Cost Type 'num-routingcost', the sollicited ALTO Server has defined 3 different daily patterns each represented by a Calendar, to cover the week of Monday June 30th at 00:00 to Sunday July 6th 23:59:

- C1 for Monday, Tuesday, Wednesday, Thursday, (week days)

- C2 for Saturday, Sunday, (week end)

- C3 for Friday (maintenance outage on July 4, 2014 from 02:00:00 GMT to 04:00:00 GMT, or big holiday such as New Year evening).

In the following example, the ALTO Client sends its request on Tuesday July 1st 2014 at 13:15.

```
POST /calendar/endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: [TODO]
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

{
  "cost-type" : {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
  "calendared" : [true],
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv4:203.0.113.45",
      "ipv6:2000::1:2345:6789:abcd"
    ]
  }
}
```

```
HTTP/1.1 200 OK
Content-Length: [TODO]
Content-Type: application/alto-endpointcost+json

{
  "meta" : {
    "cost-type" : {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
    "calendar-response-attributes" : [
       { "calendar-start-time" : Mon, 30 Jun 2014 00:00:00 GMT,
         "time-interval-size" : "1 hour",
         "numb-intervals" : 24,
         "repeated": 4 }
      ],
    } // end meta

    "endpoint-cost-map" : {
      "ipv4:192.0.2.2": {
        "ipv4:192.0.2.89"    : [v1, v2, ... v24],
        "ipv4:198.51.100.34" : [v1, v2, ... v24],
        "ipv4:203.0.113.45"  : [v1, v2, ... v24],
        "ipv6:2000::1:2345:6789:abcd" : [v1, v2, ... v24]
      }
    }
}
```

When the Client gets the Calendar for "routingcost", it sees that the "calendar-start-time" is Monday at 00h00 GMT and member "repeated" is equal to ’4’. It understands that the provides values are valid until Thursday included and will only need to get a Calendar update on Friday.

4.2.4. Example transaction for the ECS with a calendar for both routingcost and latency

In this example, it is assumed that the ALTO Server implements multi-cost capabilities, as specified in [draft-ietf-alto-multi-cost] . That is, an ALTO client can request and receive values for several cost types in one single transaction. An illustrating use case is a path selection done on the basis of 2 metrics: routing cost and latency.

As in the previous example, the IRD indicates that the ALTO Server provides "routingcost" Calendars in terms of 24 time intervals of 1 hour each.

For metric "latency", the IRD indicates that the ALTO Server provides Calendars in terms of 12 time intervals values lasting each 5 minutes.

In the following example transaction, the ALTO Client sends its request on Tuesday July 1st 2014 at 13:15.

```
POST calendar/endpointcost/lookup HTTP/1.1
  Host: alto.example.com
  Content-Length: [TODO]
  Content-Type: application/alto-endpointcostparams+json
  Accept: application/alto-endpointcost+json,application/alto-error+json

  {
    "multi-cost-types" : [
        {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
        {"cost-mode" : "numerical", "cost-metric" : "latency"}
    ],
    "calendared" : [true, true],
    "endpoints" : {
      "srcs": [ "ipv4:192.0.2.2" ],
      "dsts": [
        "ipv4:192.0.2.89",
        "ipv4:198.51.100.34",
        "ipv4:203.0.113.45",
        "ipv6:2000::1:2345:6789:abcd"
      ]
    }
```

```
  }


  HTTP/1.1 200 OK
  Content-Length: [TODO]
  Content-Type: application/alto-endpointcost+json

  {
    "meta" : {
      "multi-cost-types" : [
        {"cost-mode" : "numerical", "cost-metric" : "routingcost"},
        {"cost-mode" : "numerical", "cost-metric" : "latency"}
       ],
      "calendar-response-attributes" : [
         { "cost-type-name : num-routingcost"
           "calendar-start-time" : Mon, 30 Jun 2014 00:00:00 GMT,
           "time-interval-size" : "1 hour",
           "numb-intervals" : 24,
           "repeated": 4 },
         { "cost-type-name : num-latency"
           "calendar-start-time" : Tue, 1 Jul 2014 13:00:00 GMT,
           "time-interval-size" : "5 minute",
           "numb-intervals" : 12}
       ],
      } // end meta

      "endpoint-cost-map" : {
        "ipv4:192.0.2.2": {
          "ipv4:192.0.2.89"    : [[r1, r2, ... r24], [l1, l2, ... l12]],
          "ipv4:198.51.100.34" : [[r1, r2, ... r24], [l1, l2, ... l12]],
          "ipv4:203.0.113.45"  : [[r1, r2, ... r24], [l1, l2, ... l12]],
          "ipv6:2000::1:2345:6789:abcd" : [[r1, r2, ... r24], [l1, l2, ... l12]]
        }
      }
  }
```

When receiving the response, the client sees that the calendar values for ’routing cost’ are repeated for 4 iterations. Therefore, in its next requests until the routing cost calendar is expected to change, the client will only need to request a calendar for "latency".

Without the ALTO Calendar extensions, the ALTO client would have no clue on the dynamicity of the metric value change and would spend needless time requesting values at an inappropriate pace. In addition, without the Multi-Cost ALTO capabilities, the ALTO client would duplicate this waste of time as it would need to send one request per cost metric.

4.3. Recap of rules related to ALTO Cost Calendars

XXXXX TO BE COMPLETED + MOVED AT THE END OF THE SPECS

A Calendar-aware ALTO Server MUST implement the base protocol specified in RFC7285.

If no Calendar attributes are defined for a given Cost Type, in a given resource entry, the ALTO Server MUST set the value in the 'calendar-attributes' array to the symbol 'null' .

When a metric is available as a calendar, it MUST be available as a single value. An ALTO Server aquiring cost values in limited time intervals only can construct a single value from the value array.

Calendared information resources MUST be requested via a POST method.

If this member "repeat-indication" is not present in the calendar attributes indicated in the IRD, it MUST be assumed to have a value equal to "false".

5. Use cases for ALTO Cost Schedule

[THIS SECTION NEEDS TO BE SHORTENED]

This section presents use cases showing the benefits of ALTO Cost calendars for applications needing to decide both "where" to connect and "when".

5.1. Bulk Data Transfer scheduling upon bandwidth calendars

Large Internet Content Providers (ICPs) like Facebook or YouTube, as well as CDNs rely on data replication across multiple sites and time zones to offload the core site and increase user experience through shorter latency from a local site. Typically the usage pattern of these data centers or caches follows a location dependent diurnal demand pattern. In these examples, data replication across the various locations of an ICP, leads to bulk data transfers between datacenters on a diurnal pattern.

In the meantime, there is a degree of freedom on when the content is transmitted from the origin server to the caching node, or from the core site to a local site. However, scheduling these data transfers is a non-trivial task as they should not infer with the user peak demand to avoid degradation of user experience and to decrease billing costs for the datacenter operator by leveraging off-peak hours for the transfer.

As a result, these ICPs need to have a good knowledge on the link utilization patterns between the different datacenters before making an efficient scheduling decision. While usage data today is already gathered and used to schedule data transfers, provisioning these data gets increasingly complex with the number of CDN nodes and datacenter operators that are involved. In particular, privacy concerns prevent that this kind of data is shared across administrative domains. The ALTO Cost Calendar avoids these problems by presenting an abstracted view of time sensitive utilization maps through a dedicated ALTO service to allow ICPs a coherent scheduling of data transfers across administrative domains and time zones.

Likewise, bandwidth Calendaring allows network operators to reserve resources in advance according to agreements with their customers, enabling them to transmit data with specified starting time and duration, for example, for a scheduled bulk data replication between data centers. Traditionally, this can be supported by a Network Management System operation such as path pre-establishment and activation on the agreed starting time. However, this does not provide efficient network usage since the established paths exclude the possibility of being used by other services even when they are not used for undertaking any service.

An ALTO Cost calendar for TE metrics on transfer paths can support the scheduled bulk data replication with better efficiency since it can alleviate the processing burden on network elements.

Cost calendars for these time-sensitive ALTO TE metrics need to consider the network topology and the dynamicity of the traffic. For example, a small topology with low density and low capacity that carries inpredictable, heavy and bursty traffic has few chances to exhibit stationary TE metric value patterns over large periods and would benefit to use the ALTO Calendar over smaller time slots. Some ALTO TE metric values, even aggregated over time may need to be updated at a frequency that would require doing ALTO requests at a pace that would be overload both the ALTO Client and the Server. Large high capacity topologies would benefit from Cost Calendars with a coarse time granularity for the filtered cost map service where as Calendars of finer time granularity for the Endpoint Cost Service would be better suited for small low density and capacity topologies.

5.1.1. Applicable example transaction

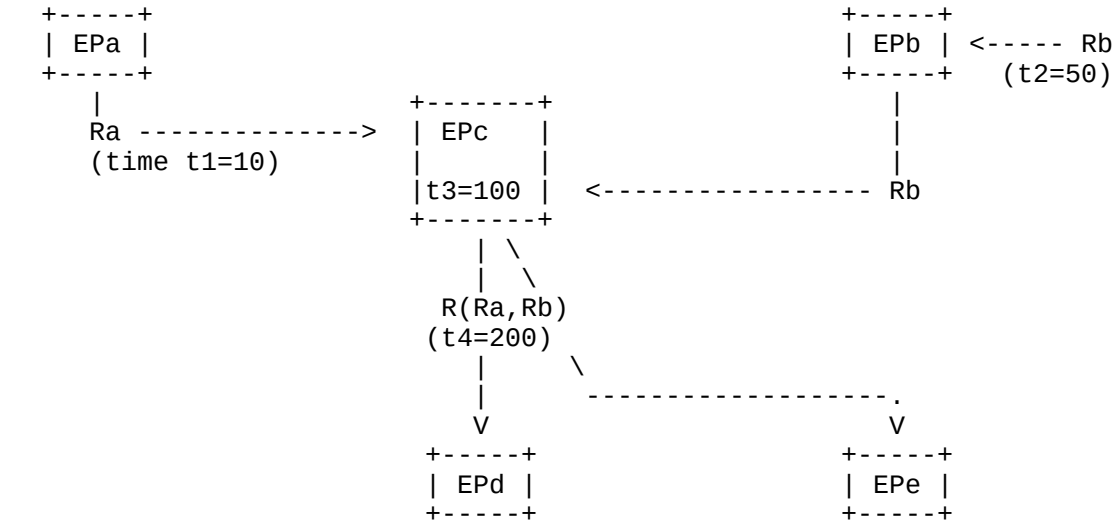
Assuming a Large high capacity topology, an applicable example transaction for this us case is provided by section 4.1.3. "Example transaction for a FCM with a "request-date" bandwidth Calendar".

5.2. Applications with limited connectivity or access to datacenters

Some applications are limited in their connectivity either in time or resources or both. For example applications running on devices in remote locations or in developing countries that need to synchronize their state with a data center periodically, in particular if sometimes there is no connection at all. Example applications are enterprise database update, remote learning, remote computation distributed on several data center endpoints.

Wireless connections have a variable quality and may even be intermittent. On the other hand, the wireless network conditions have a rapid impact on applications while they can sometimes be predicted over a span of time. Non real time applications and time-insensitive data transfers such as client patching, archive syncing, etc. can benefit from careful scheduling. It is thus desirable to provide ALTO clients with routing costs to connection nodes (i.e. Application Endpoints) over different time periods. This would allow end systems using ALTO aware application clients to schedule their connections to application endpoints.

Another challenge arises with applications using data and physical resources scattered around the world. For non-real time applications, the interaction with Endpoints can be orchestratrated and scheduled at the time slots corresponding to the best possible network conditions. For instance, resource Ra downloaded from Endpoint EPa at time t1, Resource Rb uploaded to EPb at time t2, some batch computation involving Ra and Rb done on EPc at time t3 and results R(A,B) downloaded to EPd and EPe at time t4.



5.2.1. Applicable example transaction

An applicable example transaction for this use case is provided by section 4.2.3. "Example transaction for the ECS with a "periodic" routingcost Calendar".

5.3. SDN Controller guided traffic scheduling with Calendars

An ALTO Server can assist an SDN Controller by hosting abstracted network information that can be provided to SDN aware applications via an ALTO Client.

Via the Northbound interface (NBI), applications may get QoE impacting information such as network provider preferences w.r.t. delay and bandwidth on the network paths. Such information may be provided via the ALTO Service.

One key objective of an SDN controller is the ability to balance the application traffic whenever possible. Resources availability may often be predicted and strong incentives for applications to time shift their traffic may be given by network operators appropriately setting routing cost values at different time values, according to their policy on network utilization over time.

To achieve this objective, the SDN controller can:

1. get the network state information from its controlled network elements through its southbound API and derive an estimation of these values over given time frames

2. abstract the network topology and end to end path costs and store them in an ALTO Server as Network Maps and Cost Calendars

3. deliver these values to ALTO Clients linked to SDN applications, through the NBI.

This way:

- On one hand, the applications get the best possible QoE, as they can pick the best time for them to access one or more Endpoints or PIDs,

- One the other hand, the SDN controller achieves load balancing and optimizes application traffic as it may guide the application traffic so as to better distribute the traffic over time.

5.3.1. Applicable example transaction

An applicable example transaction for this use case is provided by section 4.2.4. "Example transaction for the ECS with a calendar on both routingcost and latency".

6. IANA Considerations

Information for the ALTO Endpoint property registry maintained by the IANA and related to the new Endpoints supported by the acting ALTO server. These definitions will be formulated according to the syntax defined in Section on "ALTO Endpoint Property Registry" of [RFC7285]
,

Information for the ALTO Cost Type Registry maintained by the IANA and related to the new Cost Types supported by the acting ALTO server. These definitions will be formulated according to the syntax defined in Section on "ALTO Cost Type Registry" of [RFC7285],

6.1. Information for IANA on proposed Cost Types

When a new ALTO Cost Type is defined, accepted by the ALTO working group and requests for IANA registration MUST include the following information, detailed in Section 11.2: Identifier, Intended Semantics, Security Considerations.

6.2. Information for IANA on proposed Endpoint Propeeries

Likewise, an ALTO Endpoint Property Registry could serve the same purposes as the ALTO Cost Type registry. Application to IANA registration for Endpoint Properties would follow a similar process.

7. Acknowledgements

Thank you to Diego Lopez, He Peng and Haibin Song and the ALTO WG for fruitful discussions.

8. References

8.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

[RFC5693] Seedorf, J. and E. Burger, "Application-Layer Traffic Optimization (ALTO) Problem Statement", RFC 5693, DOI 10.17487/RFC5693, October 2009, <http://www.rfc-editor.org/info/rfc5693>.

8.2. Informative References

[draft-ietf-alto-multi-cost]
S. Randriamasy, W. Roome, N. Schwan, , "Multi-Cost ALTO (work in progress), draft-ietf-alto-multi-cost", May 2015.

[draft-wu-alto-te-metrics]
Q. Wu, Y. Yang, Y. Lee, D. Dhody, S. Randriamasy, , "ALTO Traffic Engineering Cost Metrics (work in progress)", October 2014.

[draft-yang-alto-topology]
Y. Yang, , "ALTO Topology Considerations (work in progress)", July 2013.

[ID-alto-protocol]
R.Alimi, R. Penno, Y. Yang, Eds., "ALTO Protocol, RFC 7285", September 2014.

[RFC7285] R. Alimi, R. Yang, R. Penno, Eds., "ALTO Protocol", September 2014.

[sdnrg] "Software Defined Network Research Group, http://trac.tools.ietf.org/group/irtf/trac/wiki/sdnrg".

[slides-88-alto-5-topology]
G. Bernstein, Y. Lee, Y. Yang, , , "ALTO Topology Service: Use Cases, Requirements and Framework (presentation slides IETF88 ALTO WG session), http://tools.ietf.org/agenda/88/slides/slides-88-alto-5.pdf", November 2013.

Authors' Addresses

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
NOZAY 91460
FRANCE

Email: Sabine.Randriamasy@nokia-bell-labs.com

Richard Yang
Yale University
51 Prospect st
New Haven, CT 06520
USA

Email: yry@cs.yale.edu

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

Email: sunseawq@huawei.com

Lingli Deng
China Mobile
China

Email: denglingli@chinamobile.com

Nico Schwan
Thales Deutschland

Email: nico.schwan@thalesgroup.com

ALTO Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 8, 2017

Q. Wu
Huawei
Y. Yang
Yale University
Y. Lee
D. Dhody
Huawei
S. Randriamasy
Nokia Bell Labs

July 7, 2016

ALTO Performance Cost Metrics
draft-wu-alto-te-metrics-08

Abstract

Cost Metric is a basic concept in Application-Layer Traffic Optimization (ALTO). It is used in both the Cost Map Service and the Endpoint Cost Service. Future extensions to ALTO may also use Cost Metric.

Different applications may benefit from different Cost Metrics. For example, a Resource Consumer may prefer Resource Providers that have low delay to the Resource Consumer. However the base ALTO protocol [ALTO] has defined only a single cost metric, i.e., the generic "routingcost" metric (Sec. 14.2 of ALTO base specification [ALTO]).

In this document, we proposes a set of Cost Metrics, derived and aggregated from routing protocols with different granularity and scope, such as BGP-LS,OSPF-TE and ISIS-TE, or from end to end traffic management tool. We currently define 11 new Performance Metric to measure network delay, jitter, packet loss, hop count, and bandwidth. The metrics defined in this document provide a relatively comprehensive set of Cost Metrics for ALTO and allow applications to determine "where" to connect based on end to end network performance criteria. Additional Cost Metrics such as financial cost metrics may be defined in other documents.

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

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Cost Metric is a basic concept in Application-Layer Traffic Optimization (ALTO). It is used in both the Cost Map Service and the Endpoint Cost Service. In particular, applications may benefit from knowing network performance measured on several Cost Metrics. For example, a more delay sensitive application may focus on latency, and a more bandwidth-sensitive application may focus on available bandwidth.

The objective of this document is to introduce 11 new performance cost metrics, listed in Table 1, to support the aforementioned applications and allow applications to determine "where" to connect based on end to end network performance criteria. Hence, this document extends the base ALTO protocol [ALTO], which defines only a single cost metric, i.e., the generic "routingcost" metric (Sec. 14.2 of ALTO base specification [ALTO]).

Namespace	Property	Reference
	delay	[RFCxxxx], Section 3
	delayjitter	[RFCxxxx], Section 4
	pktloss	[RFCxxxx], Section 5
	hopcount	[RFCxxxx], Section 6
	bandwidth	[RFCxxxx], Section 7
	maxbw	[RFCxxxx], Section 8
	maxresbw	[RFCxxxx], Section 9
	unresdbw	[RFCxxxx], Section 10
	residbw	[RFCxxxx], Section 11
	availbw	[RFCxxxx], Section 12
	utilbw	[RFCxxxx], Section 13

Table 1.

An ALTO server may provide a subset of the cost metrics defined in this document. These cost metrics can be retreived and aggregated from routing protocol or other traffic measurement management tool (See Figure 1).

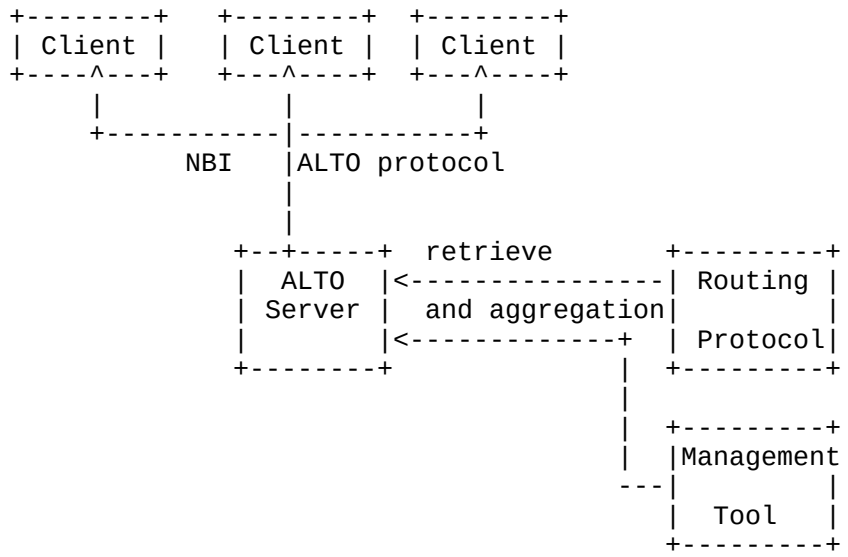


Figure 1.End to End Path Cost Metrics Exposing

When an ALTO server supports a cost metric defined in this document, the server SHOULD announce the metric in its IRD.

The definitions of a set of cost metrics can allow us to extend the ALTO base protocol (e.g., allowing output and constraints use different cost metrics), but such extensions are not in the scope of this document.

One challenge in describing the metrics is that performance metrics often depend on configuration parameters. For example, the value of packet loss rate depends on the measurement interval and varies over time. To handle this issue, ALTO server may collect data on time periods covering the past, present or only collect data on present time. The ALTO may further aggregate these data to provide an abstract and unified view that can be more useful to applications. To make the ALTO client understand whether the performance data is past data or present data, the ALTO server needs to expose to the client the validity period of each performance metric.

Following the ALTO base protocol, this document uses JSON to specify the value type of each defined metric. See [RFC4627] for JSON data type specification.

2. Data sources, computation of defined cost metrics

The cost metrics described in this document are similar, in that they may use similar data sources and have similar issues in their calculation. Hence, instead of specifying such issues for each metric individually, we specify the common issue in this section.

2.1. Data sources

An ALTO server needs data sources to compute the cost metrics described in this document. This document does not define the exact data sources. For example, the ALTO server may use log servers or the OAM system as its data source [ALTO-DEPLOYMENT]. In particular, the cost metrics defined in this document can be computed using routing systems as the data sources. Mechanisms defined in [RFC3630], [RFC3784], [OSPF-TE], [ISIS-TE], [BGP-LS] and [BGP-PM] that allow an ALTO Server to retrieve and derive the necessary information to compute the metrics that we described in this document.

2.2. Computation of metrics

An ALTO server processes measurements from data sources to compute exposed metrics. It may need performance data processing tasks such as aggregating the results across multiple systems, removing outliers, and creating additional statistics.

One specific challenge in deriving the metrics in this document is that these performance metrics depend on some configuration parameters. For example, the value of packet loss rate depends on the measurement interval and varies over time. If the ALTO server uses aforementioned routing protocol based mechanisms as data sources, then the measurement interval may be preconfigured by the routing protocol. For example, Section 5 of [ISIS-TE] defines a default measurement interval of 30 seconds. This document uses the term Measurement Interval to refer to the measurement interval used by the data sources. In the [ISIS-TE] case, it is a measurement interval set by routing protocol. The Measurement Interval(s) of the data sources can be different from the interval that this document derives the metric, e.g., the interval used by this document is multiple of measurement interval of the data sources. Hence, an ALTO server needs to resolve the mismatch, when it happens.

Another issue of converting from data source measurements to ALTO exposed metric values is that the measurement results that the ALTO Server retrieves may be defined for only links, and hence, the server will need to compose the link metrics to obtain path metrics used in services such as the Cost Map Service. In this definition, we define

the metrics to be independent of link or path, considering that future ALTO extensions may define link-based services, and hence the defined metrics should still be usable.

3. Cost Metric: Delay

Metric name:

Delay

Metric Description:

To specify spatial and temporal aggregated delay between the specified source and destination or the time that the packet spends to travel from source to destination. The spatial aggregation unit is specified in the query context (e.g., PID to PID, or endpoint to endpoint); and the temporal unit is specified as the measurement interval in the query context.

Method of Measurement or Calculation:

See section 2.2, Computation of metrics.

Units of Measurement:

The unit is microsecond.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 2.1, second paragraph for Measurement Timing.

Use and Applications:

This is intended to be a constraint attribute value. A Cost Mode is encoded as a US-ASCII string. The Metric value Type is a single 'JSONNumber' type value containing a non-negative integer component that may be followed by an exponent part.

This metric could be used as a cost metric constraint attribute used either together with cost metric attribute 'routingcost' or on its own or as a returned cost metric in the response.

Example 1: Delay value on source-destination endpoint pairs

```
POST /endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: TBA
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

{
  "cost-type": {"cost-mode" : "numerical",
                "cost-metric" : "delay"},
  "endpoints" : {
    "srcs": [ "ipv4:192.0.2.2" ],
    "dsts": [
      "ipv4:192.0.2.89",
      "ipv4:198.51.100.34",
      "ipv6:2000::1:2345:6789:abcd"
    ]
  }
}

HTTP/1.1 200 OK
Content-Length: TBA
Content-Type: application/alto-endpointcost+json
{
  "meta" :{
    "cost-type": {"cost-mode" : "numerical",
                  "cost-metric" : "delay"
     }
   },
    "endpoint-cost-map" : {
      "ipv4:192.0.2.2": {
        "ipv4:192.0.2.89"    : 10,
        "ipv4:198.51.100.34" : 20,
        "ipv6:2000::1:2345:6789:abcd"  : 30,
    }
  }
}
```

4. Cost Metric: Delay Jitter

Metric name:

Delay jitter

Metric Description:

To specify spatial and temporal aggregated jitter (latency variation) over the specified source and destination. The spatial aggregation unit is specified in the query context (e.g., PID to PID, or endpoint to endpoint); and the temporal unit is specified as the measurement interval in the query context.

Method of Measurement or Calculation:

See section 2.2, Computation of metrics.

Units of Measurement:

The unit is microsecond.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 2.1, second paragraph for Measurement Timing.

Measurement Timing:Use and Applications:

See section 3 for use and application.

```
  Example 2: Delayjitter value on source-destination endpoint pairs
  POST /endpointcost/lookup HTTP/1.1
  Host: alto.example.com
  Content-Length: TBA
  Content-Type: application/alto-endpointcostparams+json
  Accept: application/alto-endpointcost+json,application/alto-error+json

  {
    "cost-type": {"cost-mode" : "numerical",
     "cost-metric" : "delayjitter"},
    "endpoints" : {
      "srcs": [ "ipv4:192.0.2.2" ],
      "dsts": [
        "ipv4:192.0.2.89",
        "ipv4:198.51.100.34",
        "ipv6:2000::1:2345:6789:abcd"
      ]
    }
  }
  HTTP/1.1 200 OK
   Content-Length: TBA
   Content-Type: application/alto-endpointcost+json
  {
    "meta": {
             "cost type": {
             "cost-mode": "numerical",
             "cost-metric":"delayjitter"
      }
     },
    "endpoint-cost-map": {
             "ipv4:192.0.2.2": {
             "ipv4:192.0.2.89"    : 0
             "ipv4:198.51.100.34" : 1
             "ipv6:2000::1:2345:6789:abcd"  : 5
           }
        }
     }
```

5. Cost Metric: Packet Loss

Metric name:

Packet loss

Metric Description:

To specify spatial and temporal aggregated packet loss over the specified source and destination. The spatial aggregation unit is specified in the query context (e.g., PID to PID, or endpoint to endpoint); and the temporal unit is specified as the measurement interval in the query context.

Method of Measurement or Calculation:

See section 2.2, Computation of metrics.

Units of Measurement:

The unit is percentile.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 2.1, second paragraph for Measurement Timing.

Use and Applications:

See section 3 for use and application.

```
  Example 3: pktloss value on source-destination endpoint pairs
  POST /endpointcost/lookup HTTP/1.1
  Host: alto.example.com
  Content-Length: TBA
  Content-Type: application/alto-endpointcostparams+json
  Accept: application/alto-endpointcost+json,application/alto-error+json

    {
      "cost-type": {"cost-mode" : "numerical",
       "cost-metric" : "pktloss"},
      "endpoints" : {
        "srcs": [ "ipv4:192.0.2.2" ],
        "dsts": [
          "ipv4:192.0.2.89",
          "ipv4:198.51.100.34",
          "ipv6:2000::1:2345:6789:abcd"
        ]
      }
    }
  HTTP/1.1 200 OK
  Content-Length: TBA
  Content-Type: application/alto-endpointcost+json
  {
      "meta": {
                  "cost type": {
                 "cost-mode": "numerical",
                 "cost-metric":"pktloss"}
         }
      },
     "endpoint-cost-map": {
             "ipv4:192.0.2.2": {
             "ipv4:192.0.2.89"   : 0,
             "ipv4:198.51.100.34": 1,
             "ipv6:2000::1:2345:6789:abcd" : 2,
                                 }
               }
   }
```

6. Cost Metric: Hop Count

The metric hopcount is mentioned in [ALTO] as an example. This section further clarifies its properties.

Metric name:

Hop count

Metric Description:

To specify the number of hops in the path between the source endpoint and the destination endpoint. The hop count is a basic measurement of distance in a network and can be exposed as Router Hops, IP hops or other hops in direct relation to the routing prtocols originating this information. it might also result from the aggregation of such information.

Method of Measurement or Calculation:

See section 2.2, Computation of metrics.

Units of Measurement:

The unit is integer number.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 2.1, second paragraph for Measurement Timing.

Use and Applications:

See section 3 for use and application.

```
  Example 4: hopcount value on source-destination endpoint pairs
  POST /endpointcost/lookup HTTP/1.1
  Host: alto.example.com
  Content-Length: TBA
  Content-Type: application/alto-endpointcostparams+json
  Accept: application/alto-endpointcost+json,application/alto-error+json

    {
      "cost-type": {"cost-mode" : "numerical",
       "cost-metric" : "hopcount"},
      "endpoints" : {
        "srcs": [ "ipv4:192.0.2.2" ],
        "dsts": [
          "ipv4:192.0.2.89",
          "ipv4:198.51.100.34",
          "ipv6:2000::1:2345:6789:abcd"
        ]
      }
    }
  HTTP/1.1 200 OK
  Content-Length: TBA
  Content-Type: application/alto-endpointcost+json
  {
      "meta": {
                  "cost type": {
                 "cost-mode": "numerical",
                 "cost-metric":"hopcount"}
         }
      },
     "endpoint-cost-map": {
             "ipv4:192.0.2.2": {
             "ipv4:192.0.2.89"   : 5,
             "ipv4:198.51.100.34": 3,
             "ipv6:2000::1:2345:6789:abcd" : 2,
                                 }
               }
   }
```

7. Cost Metric: Bandwidth

Metric name:

Bandwidth

Metric Description:

To specify spatial and temporal aggregated bandwidth over the specified source and destination. The spatial aggregation unit is specified in the query context (e.g., PID to PID, or endhost to endhost); and the temporal unit is specified as the measurement interval in the query context.

This is just a definition of a class of cost metric 'bandwidth'. The use of this cost metric is always in conjunction with what it represents, which could be Max Bandwidth (maxbw), Residual Bandwidth (residuebw) etc.

Method of Measurement or Calculation:

See section 2.2, Computation of metrics.

Units of Measurement:

The units are bytes per second.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 2.1, second paragraph for Measurement Timing.

Use and Applications:

See section 3 for use and application.

8. Cost Metric: Maximum Bandwidth

Metric name:

Maximum Bandwidth

Metric Description:

To specify spatial and temporal maximum bandwidth over the specified source and destination. The values correspond to the

maximum bandwidth that can be used (motivated from RFC 3630 Sec. 2.5.6.). The spatial aggregation unit is specified in the query context (e.g., PID to PID, or endhost to endhost); and the temporal unit is specified as the measurement interval in the query context.

Method of Measurement or Calculation:

See section 2.2, Computation of metrics.

Units of Measurement:

See definition for the Bandwidth Cost Metric.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 2.1, second paragraph for Measurement Timing.

Use and Applications:

See section 3 for use and application.

Example 5: maxbw value on source-destination endpoint pairs

```
POST/ endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: TBA
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

{
   "cost-type": { "cost-mode":  "numerical",
   "cost-metric":  "maxbw"},
   "endpoints":  {
      "srcs": [ "ipv4 : 192.0.2.2" ],
      "dsts": [
        "ipv4:192.0.2.89",
        "ipv4:198.51.100.34",
        "ipv6:2000::1:2345:6789:abcd"
      ]
    }
  }

HTTP/1.1 200 OK
Content-Length: TBA
Content-Type: application/alto-endpointcost+json
{
    "meta": {
           "cost-type": {
           "cost-mode": "numerical",
           "cost-metric": "maxbw"
           }
    },
"endpoint-cost-map": {
          "ipv4:192.0.2.2": {
          "ipv4:192.0.2.89":    0,
          "ipv4:198.51.100.34" : 2000,
          "ipv6:2000::1:2345:6789:abcd":  5000,
                       }
        }
}
```

9. Cost Metric: Maximum Reservable Bandwidth

Metric name:

Maximum Reservable Bandwidth

Metric Description:

To specify spatial and temporal maximum reservable bandwidth over the specified source and destination. The value is corresponding to the maximum bandwidth that can be reserved (motivated from RFC 3630 Sec. 2.5.7.). The spatial aggregation unit is specified in the query context (e.g., PID to PID, or endpoint to endpoint); and the temporal unit is specified as the measurement interval in the query context.

Method of Measurement or Calculation:

See section 2.2, Computation of metrics.

Units of Measurement:

See definition of the Bandwidth Cost Metric.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 2.1, second paragraph for Measurement Timing.

Use and Applications:

See section 3 for use and application.

Example 6: maxresbw value on source-destination endpoint pairs

```
POST/ endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: TBA
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

  {
    "cost-type" { "cost-mode":  "numerical",
    "cost-metric":  "maxresbw"},
    "endpoints":  {
      "srcs": [ "ipv4 : 192.0.2.2" ],
      "dsts": [
        "ipv4:192.0.2.89",
        "ipv4:198.51.100.34",
        "ipv6:2000::1:2345:6789:abcd"
      ]
    }
  }
HTTP/1.1 200 OK
Content-Length: TBA
Content-Type: application/alto-endpointcost+json
{
    "meta": {
           "cost-type": {
           "cost-mode": "numerical",
           "cost-metric": "maxresbw"
           }
    },
  " endpoint-cost-map": {
         "ipv4:192.0.2.2" {
         "ipv4:192.0.2.89" :    0,
         "ipv4:198.51.100.34": 2000,
         "ipv6:2000::1:2345:6789:abcd":  5000,
                           }
          }
}
```

10. Cost Metric: Unreserved Bandwidth

Metric name:

Unreserved Bandwidth

Metric Description:

To specify spatial and temporal unreserved bandwidth over the specified source and destination. The values correspond to the bandwidth that can be reserved with a setup priority of 0 through 7. Therefore this metric is endcoded as an array of 8 values. The spatial aggregation unit is specified in the query context (e.g., PID to PID, or endpoint to endpoint); and the temporal unit is specified as the measurement interval in the query context.

Method of Measurement or Calculation:

See section 2.2, Computation of metrics.

Units of Measurement:

See definition for the bandwidth Cost Metric.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 2.1, second paragraph for Measurement Timing.

Use and Applications:

See section 3 for use and application.

Example 7: unresbw value on source-destination endpoint pairs
In this example, the Collection method specifies that the 'unresbw' values are defined as the 'unavailable bandwidth' specified in section 2.5.8 of RFC3630: 8 unavailable bandwidth value are reported in the same OSPF message using the same TLV. Each value is corresponding to the bandwidth that can be reserved with a setup priority of 0 through 7.

```
POST/ endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: TBA
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

  {
   "cost-type" { "cost-mode":  "numerical",
   "cost-metric":  "unresbw[1,8]" },
   "endpoints":  {
      "srcs": [ "ipv4:192.0.2.2" ],
      "dsts": [
        "ipv4:192.0.2.89",
        "ipv4:198.51.100.34",
        "ipv6:2000::1:2345:6789:abcd"
      ]
    }
  }

HTTP/1.1 200 OK
Content-Length: TBA
Content-Type: application/alto-endpointcost+json
{
   "meta": {
          "cost-type": {
          "cost-mode": "numerical",
          "cost-metric": "unresbw[1,8]"
        }
  },
"endpoint-cost-map" {
           "ipv4:192.0.2.2" {
           "ipv4:192.0.2.89" :   [0,0,0,0,0,0,0,0],
           "ipv4:198.51.100.34": [0,0,0,0,0,0,0,2000],
           "ipv6:2000::1:2345:6789:abcd":  [0,0,0,0,0,0,0,5000],
                        }
       }
}
```

11. Cost Metric: Residue Bandwidth

Metric name:

Residue Bandwidth

Metric Description:

To specify spatial and temporal residual bandwidth over the specified source and destination. The value is calculated by subtracting tunnel reservations from Maximum Bandwidth (motivated from [I-D. ietf-isis-te-metric-extensions], Sec.4.5.). The spatial aggregation unit is specified in the query context (e.g., PID to PID, or endpoint to endpoint); and the temporal unit is specified as the measurement interval in the query context.

Method of Measurement or Calculation:

See section 2.2, Computation of metrics.

Units of Measurement:

See definition of the general Bandwidth.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 2.1, second paragraph for Measurement Timing.

Use and Applications:

See section 3 for use and application.

Example 8: residuebw value on source-destination endpoint pairs

```
POST/ endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: TBA
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

  {
   "cost-type": { "cost-mode":  "numerical",
   "cost-metric":  "residubw"},
   "endpoints":  {
     "srcs": [ "ipv4 : 192.0.2.2" ],
     "dsts": [
       "ipv4:192.0.2.89",
       "ipv4:198.51.100.34",
       "ipv6:2000::1:2345:6789:abcd"
     ]
   }
}

HTTP/1.1 200 OK
Content-Length: TBA
Content-Type: application/alto-endpointcost+json
{
   "meta": {
          "cost-type" {
          "cost-mode": "numerical",
          "cost-metric": "residubw"
        }
    },
"endpoint-cost-map" {
         "ipv4:192.0.2.2" {
         "ipv4:192.0.2.89" :    0,
         "ipv4:198.51.100.34": 2000,
         "ipv6:2000::1:2345:6789:abcd":  5000,
                     }
        }
}
```

12. Cost Metric: Available Bandwidth

Metric name:

Available Bandwidth

Metric Description:

To specify spatial and temporal availaible bandwidth over the specified source and destination. The value is calculated by subtracting the measured bandwidth used for the actual forwarding of best effort traffic from Residue Bandwidth (motivated from [I-D. ietf-isis-te-metric-extensions], Sec.4.6.). The spatial aggregation unit is specified in the query context (e.g., PID to PID, or endpoint to endpoint); and the temporal unit is specified as the measurement interval in the query context.

Method of Measurement or Calculation:

See section 2.2, Computation of metrics.

Units of Measurement:

See definition of the general Bandwidth.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 2.1, second paragraph for Measurement Timing.

Use and Applications:

See section 3 for use and application.

Example 9: availbw value on source-destination endpoint pairs

```
POST /endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: TBA
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

  {
   "cost-type": { "cost-mode":  "numeric",
   "cost-metric":  "availbw"},
    "endpoints":  {
      "srcs": [ "ipv4 : 192.0.2.2" ],
      "dsts": [
        "ipv4:192.0.2.89",
        "ipv4:198.51.100.34",
        "ipv6:2000::1:2345:6789:abcd"
      ]
   }
     }

HTTP/1.1 200 OK
Content-Length: TBA
Content-Type: application/alto-endpointcost+json
{
   "meta": {
          "cost-type": {
          "cost-mode": "numeric",
          "cost-metric": "availbw"
         }
   },

  "endpoint-cost-map": {
           "ipv4:192.0.2.2" {
          "ipv4:192.0.2.89" : [6,5,7,8,4,10,7,6],
          "ipv4:198.51.100.34" : [7,4,6,8,5,9,6,7],
          "ipv6:2000::1:2345:6789:abcd" : [7,6,8,5,7,9,6,8],
                       }
        }
   }
```

13. Cost Metric: Utilized Bandwidth

Metric name:

Utilized Bandwidth

Metric Description:

To specify spatial and temporal utilized bandwidth over the specified source and destination. The value is corresponding to the actual measured bandwidth used for all traffic (motivated from [I-D. ietf-isis-te-metric-extensions], Sec.4.7.). The spatial aggregation unit is specified in the query context (e.g., PID to PID, or endpoint to endpoint); and the temporal unit is specified as the measurement interval in the query context.

Method of Measurement or Calculation:

See section 2.2, Computation of metrics.

Units of Measurement:

See definition of the general Bandwidth.

Measurement Point(s) with Potential Measurement Domain:

See section 2.1, Data sources.

Measurement Timing:

See section 2.1, second paragraph for Measurement Timing.

Use and Applications:

See section 3 for use and application.

Example 10: utilbw value on source-destination endpoint pairs

```
POST /endpointcost/lookup HTTP/1.1
Host: alto.example.com
Content-Length: TBA
Content-Type: application/alto-endpointcostparams+json
Accept: application/alto-endpointcost+json,application/alto-error+json

 {
  "cost-type": {"cost-mode" : "numerical",
  "cost-metric" :  "utilbw"},
  "endpoints":  {
       "srcs" : [ "ipv4 : 192.0.2.2" ],
       "dsts" : [
         "ipv4:192.0.2.89",
         "ipv4:198.51.100.34",
         "ipv6:2000::1:2345:6789:abcd"
      ]
    }
 }

HTTP/1.1 200 OK
Content-Length: TBA
Content-Type: application/alto-endpointcost+json
 {
  "meta": {
         "cost type": {
         "cost-mode": "numerical",
         "cost-metric": "utilbw"
        }
  },
"endpoint-cost-map": {
           "ipv4:192.0.2.2" {
           "ipv4:192.0.2.89" :   0,
           "ipv4:198.51.100.34" : 2000,
           "ipv6:2000::1:2345:6789:abcd" :  5000,
                        }
         }
}
```

14. Security Considerations

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

However concerns addressed in Sections "15.1 Authenticity and Integrity of ALTO Information", "15.2 Potential Undesirable Guidance

from Authenticated ALTO Information" and "15.3 Confidentiality of ALTO Information" remain of utmost importance. Indeed, TE performance is a highly sensitive ISP information and sharing TE metric values in numerical mode requires full mutual confidence between the entities managing the ALTO Server and Client. Numerical TE performance information will most likely be distributed by ALTO Servers to Clients under strict and formal mutual trust agreements. One the other hand, ALTO Clients must be cognizant on the risks attached to such information that they would have acquired outside formal conditions of mutual trust.

15. IANA Considerations

IANA has added the following entries to the ALTO cost map Properties registry, defined in Section 3 of [RFCXXX].

Namespace	Property	Reference
	delay	[RFCxxxx], Section 3
	delayjitter	[RFCxxxx], Section 4
	pktloss	[RFCxxxx], Section 5
	hopcount	[RFCxxxx], Section 6
	bandwidth	[RFCxxxx], Section 7
	maxbw	[RFCxxxx], Section 8
	maxresbw	[RFCxxxx] Section 9
	unresdbw	[RFCxxxx], Section 10
	residbw	[RFCxxxx], Section 11
	availbw	[RFCxxxx], Section 12
	utilbw	[RFCxxxx], Section 13

16. References

16.1. Normative References

[I-D.ietf-idr-te-pm-bgp]
Previdi, S., Wu, Q., Gredler, H., Ray, S., jefftant@gmail.com, j., Filsfils, C., and L. Ginsberg, "BGP-LS Advertisement of IGP Traffic Engineering Performance Metric Extensions", draft-ietf-idr-te-pm-bgp-03 (work in progress), May 2016.

[I-D.ietf-isis-te-metric-extensions]
Previdi, S., Giacalone, S., Ward, D., Drake, J., and W. Wu, "IS-IS Traffic Engineering (TE) Metric Extensions", draft-ietf-isis-te-metric-extensions-11 (work in progress), February 2016.

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

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, DOI 10.17487/RFC4627, July 2006, <http://www.rfc-editor.org/info/rfc4627>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <http://www.rfc-editor.org/info/rfc5234>.

[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, <http://www.rfc-editor.org/info/rfc7285>.

[RFC7471] Giacalone, S., Ward, D., Drake, J., Atlas, A., and S. Previdi, "OSPF Traffic Engineering (TE) Metric Extensions", RFC 7471, DOI 10.17487/RFC7471, March 2015, <http://www.rfc-editor.org/info/rfc7471>.

[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, <http://www.rfc-editor.org/info/rfc7752>.

16.2. Informative References

[I-D.ietf-alto-deployments]
Stiemerling, M., Kiesel, S., Scharf, M., Seidel, H., and S. Previdi, "ALTO Deployment Considerations", draft-ietf-alto-deployments-15 (work in progress), May 2016.

[RFC6390] Clark, A. and B. Claise, "Framework for Performance Metric Development", RFC 6390, July 2011.

Authors' Addresses

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

Email: bill.wu@huawei.com

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

Email: yry@cs.yale.edu

Young Lee
Huawei
1700 Alma Drive, Suite 500
Plano, TX 75075
USA

Email: leeyoung@huawei.com

Dhruv Dhody
Huawei
Leela Palace
Bangalore, Karnataka 560008
INDIA

Email: dhruv.ietf@gmail.com

Sabine Randriamasy
Nokia Bell Labs
Route de Villejust
Nozay 91460
FRANCE

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

ALTO WG
Internet-Draft
Intended status: Informational
Expires: January 9, 2017

Q. Xiang
Tongji/Yale University
H. Newman
California Institute of Technology
G. Bernstein
Grotto Networking
A. Mughal
J. Balcas
California Institute of Technology

July 8, 2016

Traffic Optimization for ExaScale Science Applications
draft-xiang-alto-exascale-network-optimization-00.txt

Abstract

Massive datasets continue to be acquired, simulated, processed and analyzed by globally distributed scientific collaborations, and the volume of this data is growing exponentially. These datasets need to be exchanged through a global network infrastructure. Applications that manages the transfer of such massive data volumes can benefit substantially from the utilization of network information, and more directly from network-resident services that optimize and load balance network usage among multiple transfer requests, and coordinate the network use with the use of other resources such as computing and storage.

The Application-Layer Traffic Optimization (ALTO) protocol can provide via extensions such network information, to both users and proactive network management services where applicable, with the goal of improving both application performance and network resource utilization, leading to greater overall efficiency of the science programs' workflows.

This document introduces an Exascale Dataset Transfer Orchestrator (EDTO), which is a data transfer scheduling service for exascale science networks. EDTO provides simple APIs for users to submit, update and delete data transfer requests and to monitor the status of each transfer, along with local and global network and site state information in real-time. EDTO collects network information from multiple ALTO services utilizing proposed topology extensions and leverages emerging SDN control capabilities to orchestrate the scheduling of multiple large dataset transfers, leading to improved data transfer latency and reliability as well as more efficient utilization of limited network resources.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Scientific innovation continues to exponentially increase the production of valuable research data. Exchange of this information typically involves the worldwide network infrastructure. One leading example is the Large Hadron Collider (LHC) high energy physics (HEP) program, which aims to find new particles and interactions in a previously inaccessible range of energies. The scientific collaborations that have built and operate large HEP experimental facilities at the LHC, such as the Compact Muon Solenoid (CMS) and A Toroidal LHC ApparatuS (ATLAS), currently have more than 300 petabytes of data under management at hundreds of sites around the world, and this volume is expected to grow to one exabyte by approximately 2018.

With such an increasing data volume, the management of large data transfers in a globally distributed infrastructure has become an increasingly challenging issue. Applications such as the Production ANd Distributed Analysis system (PanDA) in ATLAS and the Physics Experiment Data Export system (PhEDEX) in CMS have been developed to manage the data transfers. Given a data transfer request, these applications make data transfer decisions based on the availability of dataset replicas at different sites, and initiate retransmission from a different replica if the original transmission fails or is excessively delayed. Such transfer scheduling applications do not take network status information directly into account, leading to suboptimal data transfer performance and underutilization of limited network resources such as bandwidth.

Integrating network status information as well as flow sterring and load balancing functions into the transfer scheduling decision process can substantially improve users' experience in terms of data transfer latency and reliability, and achieve more efficient utilization of limited resources such as bandwidth in global science networks, and the associated storage at the destination sites.

This document introduces a centralized data transfer scheduling service, Exascale Dataset Transfer Orchestrator (EDTO), which provides efficient large data transfers scheduling and resource allocation for exascale science networks. EDTO provides a set of simple API for authorized users to submit, update and delete data transfer requests. It gathers network information through a series of ALTO services, including the ones defined in [RFC7285] (network map, cost map, endpoint cost, etc.) and topology extension services (network graph [DRAFT-NETGRAPH], path vector [DRAFT-PV], routing state abstraction[DRAFT-RSA], etc.), and utilize this information to make transfer scheduling decisions including dataset replica selection, routing path computation and network resource allocation.

An EDTO prototype has been implemented on a single-domain Caltech/StarLight/Michigan/Fermilab SDN development testbed, where the ALTO OpenDaylight controller is used to collect topology information. The CMS experiment is currently pursuing the pre-production deployments of EDTO which may lead towards more widespread production use. To achieve this goal, it is imperative to collect sufficient topology information from different sites of the multi-domain CMS network without causing any privacy leak. To this end, ALTO topology extension services such as the routing state abstraction service [DRAFT-RSA] are needed.

This document is organized as follows: Section 3 defines the Terminology and Notation in this document. Section 4 elaborate on the motivation and challenges for coordinating storage, computing and network resources in a globally distributed science network infrastructure. Section 5 gives the details of EDTO architecture for orchestrating exascale dataset transfer. Section 6 discusses current development progress of EDTO and next steps.

2. Requirements Language

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

3. Terminology and Notation

This document uses the following additional terms: DTT, DTS, Relation.

- Dataset

  A set of files that are grouped together. This is the minimum data transfer unit users can manipulate. See Section 5 for a more detailed description.

- DTR

  Dataset Transfer Request. A transfer request where users specify the name of dataset and the destination site name. See Section 5 for a more detailed description.

- EDTO

  Exascale Dataset Transfer Orchestrator. A designed framework for providing data transfer scheduling service considering network information, data transfer requests, and source and destination site information. See Section 5 for a more detailed description.

4. Problem Settings

4.1. Motivation

Exascale science programs usually involve the participation of countries and sites all over the world. The CMS experiment in the LHC physics program is a typical example. The site located at the LHC laboratory is called a Tier-0 site, which processes the data selected and stored locally by the online systems that select and record the data in real-time as it comes off the particle detector, archives it and transfers it to over 10 Tier-1 sites around the globe. Raw datasets and processed datasets from Tier-1 sites are then transferred to over 160 Tier-2 sites around the world based on users' requests. Different sites have different resources, and belong to different administration domains. With the exponentially increasing data volume in the CMS experiment, the management of large data transfers in such a global multi-domain science network has become an increasingly challenging issue. Scheduling different users' dataset transfer requests to different sites requires careful orchestrating as different transfer flows are competing for limited storage, computation and network resources.

4.2. Challenges

Orchestrating exascale dataset transfers in a globally distributed science network is non-trivial as it needs to cope with two challenges.

- Different sites in this network belong to different administration domain. Sharing raw network information would violate sites' privacy constraint. Orchestrating data transfers based on highly abstracted, non-real-time network information may lead to suboptimal scheduling decisions. Hence the orchestrating framework must be able to collect sufficient network information in real-time as well as over the longer term, to allow reasonably optimized network resource utilization without violating sites' privacy requirements.

- Different science programs tend to adopt different software infrastructures for managing data and transfers, and may place different requirements on data transfers. Hence the orchestrating framework must be modular so that it can support different dataset management systems and different orchestrating algorithms.

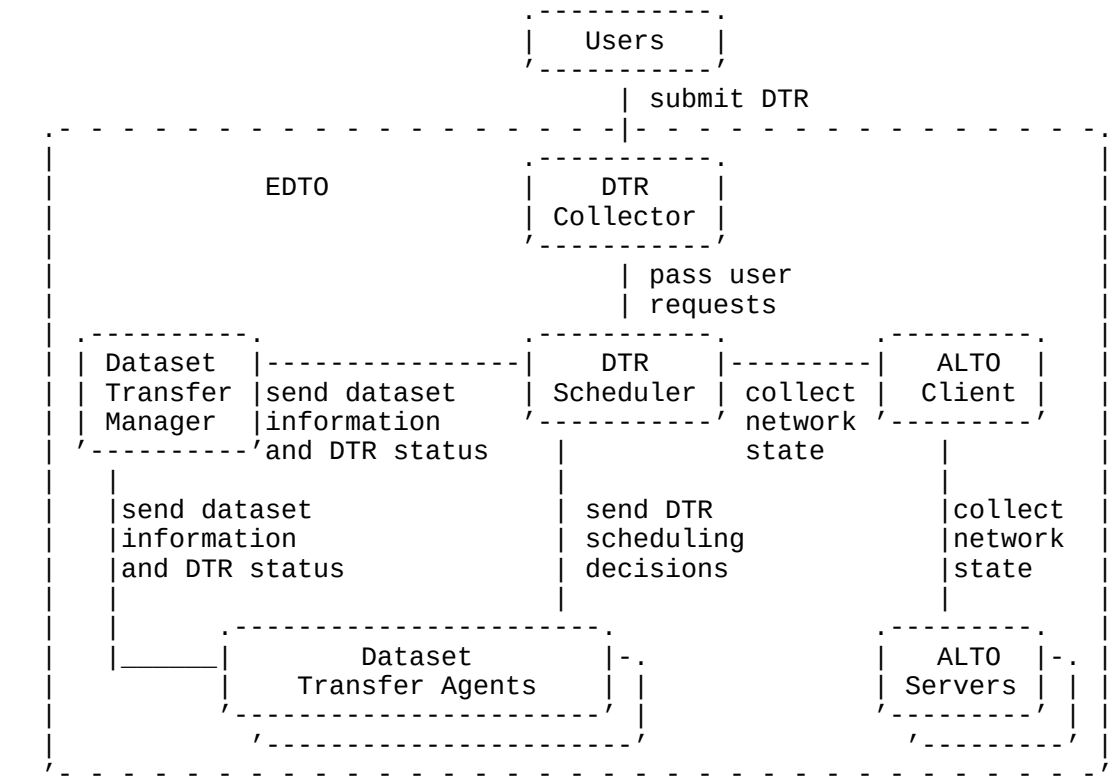
- The orchestrating framework must support the interaction between the dataset management module and the dataset transfer orchestrating module. The key information to be exchanged between modules includes dataset information, the state of the network, the flows in progress, as well as trends and network-segment and site performance from the network point of view. Such interaction ensures that (1) the various programs can adapt their data transfer management systems to be more network aware, and more efficient in achieving their goals; and (2) the various orchestrating algorithms can achieve a reasonably optimized utilization on not only the network resource, but also the computing and storage resources.

5. Exascale Dataset Transfer Orchestrator Framework

5.1. Architecture

This section describes the design details of key components of the EDTO framework: the DTR collector, ALTO client, ALTO servers, dataset transfer manager, dataset transfer agents and the DTR scheduler. Among these modules, the DTR collector provides a set of simple APIs for authorized users to submit, update and cancel dataset transfer requests. The dataset transfer manager collects ongoing DTR progress and dataset information from dataset transfer agents. The ALTO client collects network information from ALTO servers deployed at different sites.

All collected information are sent to the DTR scheduler, which makes dynamic scheduling decisions such as replica selection, routing path computation and bandwidth allocation. These decisions are then sent to data transfer agents and the network data plane for execution. Figure 1 shows the whole process.



The benefits of EDTO include:

- It provides a set of simple APIs for authorized users to submit, update and cancel dataset transfer requests, and enables real-time DTR progress monitoring.

- It improves the latency and reliability of DTR flows and achieves a better network resource allocation, by orchestrating the scheduling of all DTRs in a centralized framework that takes local, regional and global network state as well as the transfers in progress into account.

- The architecture of EDTO is modular to support different scheduler algorithms, different data transfer managers and agents, and different implementation of the ALTO servers and clients.

5.2. DTR Collector

The DTR collector is the front end of EDTO, and is responsible for collecting dataset transfer requests from users and passing them to the back end of EDTO for further processing. It provides a set of simple APIs for users to submit, update and delete transfer requests, and to track the status of dataset transfers in real-time.

5.2.1. User API

- submitDTR(datasetName, dstSite, [options])

  This API allows users to submit a DTR by specifying the name of a dataset and the destination site. It returns a request identifier requestID that allows users to update, delete this DTR or track the progress of this DTR. Users can specify additional requirements such as priority, delay, etc. when submitting the transfer request. These requirements may or may not be approved, and the relative priorities may be modified by the DTR scheduler depending on the role of users (regular users or administrators at different levels), the available network resources and dataset availability, as well as the transfer performance for each source-destination pair.

- updateDTR(requestID, [options])

  This API allows users to update the requirements of a DTR. It will return a SUCCESS if the requirements are received by the DTR scheduler. But these requirements may or may not be approved, and may be modified by the DTR scheduler depending on the role of users (regular users or administrators), the available network resources and dataset availability as well as the transfer performance for each source-destination pair.

- deleteDTR(requestID)

  This API allows users to delete a DTR by specifying the corresponding requestID. A completed DTR cannot be deleted. Ongoing DTR flows will be stopped and the transferred data will be deleted from the destination site.

- getDTRStatus(DTRID)

This API allows users to query the status of a DTR by specifying the corresponding requestID. The returned status information includes whether data transmission has started for the request, the assigned priority, the percentage of finished data transmission, transmission statistics, the expected remaining time to finish, etc.

5.3. ALTO Client

The ALTO client is in the back end of EDTO, and is responsible for retrieving network information from ALTO servers that are deployed at different sites, and in the network. The network information needed in EDTO includes the topology, link bandwidth, etc. The base ALTO protocol [RFC7285] provides an extreme single-node abstraction for this information. It is sufficient for resource allocation for a single flow. When orchestrating flows in response to multiple DTR, ALTO topology extension services such as routing state abstraction (RSA) [DRAFT-RSA], path vector [DRAFT-PV] and network graph [DRAFT-NETGRAPH] are needed to provide sufficient information for the DTR scheduler to efficiently utilize limited network information such as bandwidth.

5.3.1. Query Mode

The ALTO client should operate in different query modes depending on the implementation of ALTO servers. If an ALTO server does not support incremental updates using server-sent events (SSE) [DRAFT-SSE], the ALTO client send queries to this server periodically to get the latest network information. If the network state changes after one query, the ALTO client will not be aware of the change until next query. If an ALTO server supports SSE, the ALTO client only sends one query to the ALTO server to get the initial network information. When the network state changes, the ALTO client will be notified by the ALTO server through SSE.

5.4. ALTO Server

ALTO servers are deployed at different sites around the world, and at strategic locations in the network itself, to provide network information, including topology, link bandwidth and etc., in response to queries from the ALTO client deployed in EDTO. Each ALTO server must provide basic information services as specified in [RFC7285] such as network map, cost map, endpoint cost service (ECS), etc. To support the efficient utilization of limited network resource in EDTO, each ALTO server should also provide more fine-grained topology information through ALTO topology extension services such as the routing state abstraction [DRAFT-RSA], path vector [DRAFT-PV] and network graph [DRAFT-NETGRAPH] services.

5.5. Dataset Transfer Manager

The dataset transfer manager is responsible for the following functions:

- Collect the storage and computational resource information at different sites, and send this information to the DTR scheduler.
- Collect information about datasets at different sites, and send these information to the DTR scheduler. Such information include the volume, location and availability of every dataset, along with the recent performance history for transfers between each source-destination site pair.

The dataset transfer manager is an information source for the DTR scheduler. Different manager systems can be adopted depending on the specific system requirements. For instance, in the CMS experiment the PhEDEx transfer management database plays the role of dataset transfer manager, and in the ATLAS experiment the PanDA system plays the role.

5.6. Dataset Transfer Agents

Dataset transfer agents are deployed at each site and in the network as needed, and are responsible for the following functions:

- Receive and process instructions from the DTR scheduler, e.g. starting a new transfer, aborting a running transfer and adjusting transfer parameters such as transfer rate and number of connections.
- Monitor the status of DTRs and send the updated status to the dataset transfer manager.
- Collect the storage and computation resource information of the deployed site, and send this information to the dataset transfer manager.

Different systems can adopt different dataset transfer agents, or different structured agent subsystems, depending on specific needs. For instance, in the CMS experiment, these agents are PhEDEx distributed agents.

5.7. DTR Scheduler

The DTR Scheduler takes the dataset information collected by the dataset transfer manager, the network information collected by the ALTO client and the DTR submitted by users as input. It then makes

dataset transfer scheduling decisions, including dataset replica selection, path selection, and bandwidth allocation, for all DTRs. These decisions are sent to data transfer agents and the network data plane for execution.

5.7.1. Scheduling Algorithms

The modular design of EDTO allows the adoption of different scheduling algorithms and methodologies, depending on the specific performance requirements. In Section 5.7.3, a max-min fairness resource allocation algorithm is described, taking the CMS experiment as a use case.

5.7.2. Dynamic Scheduling

The DTR scheduler should adjust the scheduling decisions based on the history of DTR status, the utilization of different DTR flows and the network state. In normal cases, the DTR scheduler periodically collects dataset information, DTR status and network information and executes the scheduling algorithm based on the collected information. When it is notified of events such as DTR status update, network state update and etc., the DTR scheduler will also execute the scheduling algorithm to make scheduling and bandwidth allocation adjustments.

5.7.3. Example: A Max-Min Fairness Resource Allocation Algorithm

In this section, we describe a max-min fair resource allocation (MFRA) scheduling algorithm which aims to minimize the maximal time to complete a DTR subject to a set of constraints. To make resource allocation decisions, MFRA requires sufficient network information including topology, link bandwidth and recent historical information in some cases. In a small-scale single-domain network, an SDN controller can provide the raw complete topology information for the MFRA algorithm. However, in a large-scale multi-domain science network such as CMS, providing the raw network topology is infeasible because (1) it would incur significant communication overhead; and (2) it would violate the privacy constraints of some sites. Several ALTO extension topology services including Abstract Path Vector [DRAFT-PV], Network Graphs [DRAFT-NETGRAPH] and RSA [DRAFT-RSA] can provide the fine-grained yet aggregated/abstract topology information for MFRA to efficiently utilize bandwidth resources in the network.

Ongoing pre-production deployment efforts of EDTO in the CMS network involve the implementation of the RSA service. Other than topology information, the additional input of the MFRA algorithm is the priority of each class of flows, expressed in terms of upper and

lower limits on the allocated bandwidth between the source and destination for each DTR.

The basic idea of the MFRA algorithm is to iteratively maximize the volume of data that can be transferred subject to the constraints. It works in quantized time intervals such that it schedules network paths and data volumes to be transferred in each time slot. When the DTR scheduler is notified of events such as the cancellation of a DTR, the completion of a DTR or network state changes, the MFRA algorithm will also be invoked to make updated network path and bandwidth allocation decisions.

In each execution cycle, MFRA first marks all transfers as unsaturated. Then it solves a linear programming model to find the common minimum transfer satisfaction rate (i.e., the ratio of transferred data volume in a time interval over the whole data volume of this request) that is satisfied by all transfer requests. With this common rate found, MFRA then randomly selects an unsaturated request in each iteration, increases its transfer rate as much as possible by finding residual paths available in the network, or by increasing the allocated bandwidth along an existing path, until it reaches its upper limit or can otherwise not be increased further, so it is saturated. At each iteration, newly saturated requests are removed from the subsequent process by fixing their corresponding rate value, and completed transfers are removed from further consideration. After all the data transfer rates are saturated in the given time slot, then a feasible set of data transfer volumes scheduled to be transferred in the slot across each link in the network can be derived.

The MFRA algorithm yields a full utilization of limited network resources such as bandwidth so that all DTR can be completed in a timely manner. It allocates network resources fairly so that no DTR suffers starvation. It also achieves load balance among the sites and the network paths crossing a complex network topology so that no site and no network link is oversubscribed. Moreover, MFRA can handle the case where particular routing constraints are specified, e.g., where all routes are fixed ahead of time, or where each transfer request only uses one single path in each time slot, by introducing an additional set of linear constraints.

6. Discussion

6.1. Deployment

The EDTO framework is the first step towards a new class of intelligent, SDN-driven global systems for data intensive science programs involving a worldwide ensemble of sites and networks, such

as CMS and ATLAS. EDTO relies heavily on the ALTO services for collecting and expressing abstract up-to-date network information, and the SDN centralized control capability to orchestrate the flows of multiple DTRs. It aims to provide a new operational paradigm in which science programs can use complex network and computing infrastructures with high throughput, while allowing for coexistence with other network traffic.

An prototype case study implementation of EDTO has been demonstrated on the Caltech/StarLight/Michigan/Fermilab SDN development testbed. Because this testbed is a single-domain network, the current EDTO prototype leverages the ALTO OpenDaylight controller, to collect topology information. The CMS experiment is currently exploring pre-production deployments of EDTO, looking towards future widespread production use. To achieve this goal, it is imperative to collect sufficient topology information from the various sites in the multi-domain CMS network, without causing any privacy leak. To this end, the ALTO RSA service [DRAFT-RSA] is under development. Furthermore, as will be discussed next, other ALTO topology extension services can also substantially improve the performance of the DTR scheduler.

6.2. Benefiting From ALTO Extension Topology Services

The current ALTO base protocol [RFC7285] expose network topology using the extreme "my-Internet-view" representation, which abstracts a whole network as a single node that has a set of access ports, with each port connects to a set of endhosts called endpoints. Such extreme abstraction lead to significant information loss on network topology [DRAFT-PV], which is key information for EDTO to make dynamic scheduling and resource allocation decisions. Though EDTO can still schedule DTR flows on this abstract view, the scheduling and resource allocation decisions are suboptimal. Alternatively, feeding the raw, complete network topology of each site to EDTO is not desirable, either. First, this would violate privacy constraints of different sites. Secondly, a raw network topology would significantly increase the problem space and the solution space of DTR scheduler, leading to a long computation time. Hence, the DTR scheduler desires an ALTO topology service that is able to provide only enough fine grained topology information. Several ALTO topology extension services including Abstract Path Vector [DRAFT-PV], Network Graphs [DRAFT-NETGRAPH] and RSA [DRAFT-RSA] are potential candidates for improving the performance of the DTR scheduler. For instance, the path vector service supports the capacity region query, which accepts multiple concurrent data flows as the input and returns the information of bottleneck links, such as bandwidth, for the given set of concurrent flows. This information can be interpreted as a set of linear constraints for the DTR scheduler, which can help the scheduler better utilize the limited bandwidth resource.

6.3. Constraints of the MFRA Algorithm

The first constraint of the MFRA algorithm is computation overhead. The execution of MFRA involves solving linear programming problems repeatedly at every time slot. The overhead of computation time is acceptable for small sets of DTRs, but may increase significantly when handling large sets of DTRs, e.g., hundreds of transfer requests. Current efforts towards addressing this issue include exploring the feasibility of incremental computation of scheduling policies, and reducing the problem scale by finding the minimal equivalent set of constraints of the linear programming model. The latter approach can benefit substantially from the ALTO RSA service [DRAFT-RSA].

The second constraint is that the current version of MFRA does not involve dataset replica selection. Simply denoting the replica selection as a set of binary constraint will significantly increases the computation complexity of the scheduling process. Current efforts focus on finding efficient algorithms to make dataset replica selection.

6.4. EDTO Extension for Supporting Data Analysis and Processing Orchestration

The base EDTO framework focus on orchestrating multiple DTR flows in the network. In a globally distributed scientific infrastructure such as LHC, data processing and analysis applications such as MapReduce also requires transferring large amount of data between endhosts or sites. How to orchestrate the data flow of such applications through the whole network is an imperative task. The EDTO framework is expected to extend the DTR scheduler to support the joint optimization of both computing resource and network resource among different data analysis tasks.

7. Security Considerations

This document does not introduce any privacy or security issue not already present in the ALTO protocol.

8. IANA Considerations

This document does not define any new media type or introduce any new IANA consideration.

9. Acknowledgments

The authors thank discussions with Mingming Chen, Kai Gao, Xiao Lin, Xin Wang, Y. Richard Yang and Jingxuan Zhang.

10. References

10.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

10.2. Informative References

[DRAFT-NETGRAPH] Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Topology Extensions: Node-Link Graphs", 2015, <https://tools.ietf.org/html/draft-yang-alto-topology-06>.

[DRAFT-PV] Bernstein, G., Lee, Y., Roome, W., Scharf, M., and Y. Yang, "ALTO Extension: Abstract Path Vector as a Cost Mode", 2015, <https://tools.ietf.org/html/draft-yang-alto-path-vector-01>.

[DRAFT-RSA] Gao, K., Wang, X., Yang, Y., and G. Chen, "ALTO Extension: A Routing State Abstraction Service Using Declarative Equivalence", 2015, <https://datatracker.ietf.org/doc/draft-gao-alto-routing-state-abstraction/>.

[DRAFT-SSE] Roome, W. and Y. Yang, "ALTO Incremental Updates Using Server-Sent Events (SSE)", 2015, <https://datatracker.ietf.org/doc/draft-ietf-alto-incr-update-sse/>.

[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, <http://www.rfc-editor.org/info/rfc7285>.

Authors' Addresses

Qiao Xiang
Tongji/Yale University
51 Prospect Street
New Haven, CT
USA

Email: qiao.xiang@cs.yale.edu

Harvey Newman
California Institute of Technology
1200 California Blvd.
Pasadena, CA
USA

Email: newman@hep.caltech.edu

Greg Bernstein
Grotto Networking
Fremont, CA
USA

Email: gregb@grotto-networking.com

Azher Mughal
California Institute of Technology
1200 California Blvd.
Pasadena, CA
USA

Email: azher@hep.caltech.edu

Justas Balcas
California Institute of Technology
1200 California Blvd.
Pasadena, CA
USA

Email: justas.balcas@cern.ch

ALTO WG
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2017

G. Bernstein
Grotto Networking
K. Gao
Tsinghua University
Y. Lee
Huawei
W. Roome
M. Scharf
Nokia
Y. Yang
Yale University

July 8, 2016

ALTO Extension: Path Vector Cost Mode
draft-yang-alto-path-vector-03.txt

Abstract

The Application-Layer Traffic Optimization (ALTO) Service has defined network and cost maps to provide basic network information, where the cost maps allow only scalar (numerical or ordinal) cost mode values. This document introduces a new cost mode called path-vector to allow ALTO clients to support use cases such as capacity regions for applications. This document starts with a non-normative example called multi-flow scheduling (or capacity region) to illustrate that ALTO cost maps without path vectors cannot provide sufficient information. This document then defines path-vector as a new cost mode.

Requirements Language

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

Status of This Memo

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

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

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any

time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 9, 2017.

Table of Contents

1. Introduction

The ALTO base protocol [RFC7285] is designed for a setting of exposing network topology using the extreme "my-Internet-view" representation, which abstracts a whole network as a single node that has a set of access ports, with each port connects to a set of endhosts. This "single-node" abstraction is simple and can support a wide range of applications already.

A problem of this abstraction, however, is that it does not provide sufficient information for use cases that require exposure of topology information beyond the single-node abstraction, for example,

when there is a need to detect sharing of resources in the underlying topology (see an example in Section 3).

This document goes beyond the single-node topology by introducing path vector as a new ALTO cost mode, where each path vector specifies abstracted network elements on the routing paths of a set of flows. Since the network elements on a path vector are abstract network elements defined by ALTO servers, the new path-vector cost mode provides a mechanism to allow a network to control the level of topology exposure, and at the same time better support application traffic optimization. The design of path vector is based on the ALTO WG discussions at IETF 89, with summary slides at http://tools.ietf.org/agenda/89/slides/slides-89-alto-2.pdf.

The organization of this document is organized as follows. Section 2 gives a non-normative example called multi-flow scheduling to illustrate the need to introduce path vectors. Sections 3-6 specify the path vector cost mode for query, response, announcements, and incremental udpates, respectively. Sections 7 and 8 discuss security and IANA considerations.

2. Example: Capacity Region for Multi-Flow Scheduling

Consider the case that routing is given. Then what application-layer traffic optimization will focus on is traffic scheduling among application-layer paths. Specifically, assume that an application has control over a set of flows F = {f1, f2, ..., f_|F|}. If routing is given, what the application can control is x1, x2, ..., x_|F|, where xi is the amount of traffic for flow i. Let x = [x1, ..., x_|F|] be the vector of the flow traffic amounts. Due to shared links, feasible values of x where link capacities are not exceeded can be a complex polytope.

Specifically, consider a network as shown in Figure 1. The network has 7 switches (sw1 to sw7) forming a dumb-bell topology. Switches sw1/sw3 provide access on one side, s2/s4 provide access on the other side, and sw5-sw7 form the backbone. Endhosts eh1 to eh4 are connected to access switches sw1 to sw4 respectively. Assume that the bandwidth of each link is 100 Mbps.

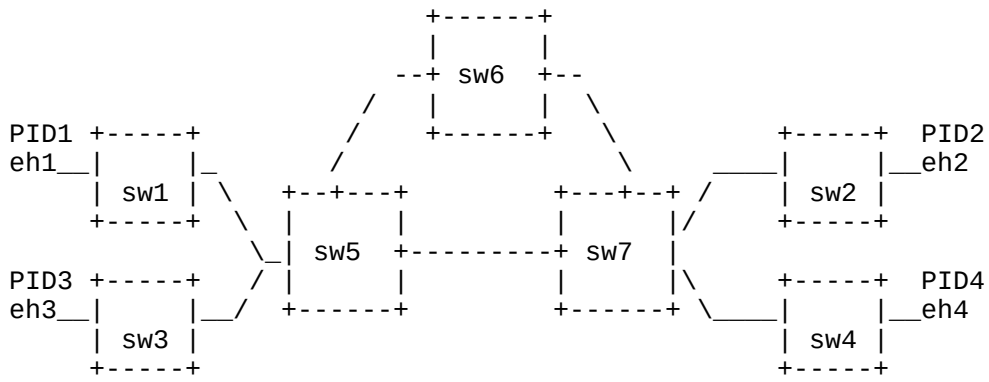


Figure 1: Raw Network Topology.

The single-node ALTO topology abstraction of the network is shown in Figure 2.

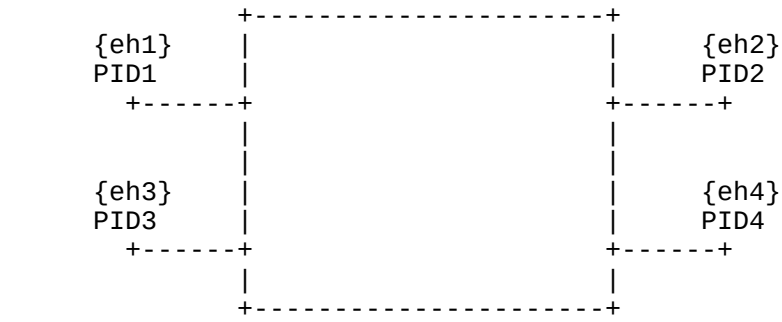


Figure 2: Base Single-Node Topology Abstraction.

Consider an application overlay (e.g., a large data analysis system) which needs to schedule the traffic among a set of endhost source-destination pairs, say eh1 -> eh2, and eh3 -> eh4. The application can request a cost map providing end-to-end available bandwidth, using 'available bw' as cost-metric and 'numerical' as cost-mode.

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

- Case 1: If eh1 -> eh2 and eh3 -> eh4 use different paths, for example, when the first uses sw1 -> sw5 -> sw7 -> sw2, and the

second uses sw3 -> sw5 -> sw6 -> sw7 -> sw4. Then the application will obtain 200 Mbps.

- Case 2: If eh1 -> eh2 and eh3 -> eh4 share a bottleneck, for example, when both use the direct link sw5 -> sw7, then the application will obtain only 100 Mbps.

To allow applications to distinguish the two aforementioned cases, the network needs to provide more details. The path vector extension defined in this document resolves this issue.

See [I-D.bernstein-alto-topo] for a survey of use-cases where extended network topology information is needed.

3. Path-Vector Query

Since the key motivation of providing path vectors is for concurrent-flow settings, due to sharing of network resources, the specification is for a multi-flow setting. Also, instead of defining such queries separately for grouping of endhosts (PIDs) and individual endpoints, we define for generic sources, destinations.

```
POST /capacityregion/lookup HTTP/1.1
Host: alto.example.com
Content-Length: TBD
Content-Type: application/alto-flowparams+json
Accept: application/alto-costmap+json,application/alto-error+json

{
  "cost-type" : {"cost-mode":   "path-vector",
                 "cost-metric": "bw"}
  },
  "flows" : [
    {"src": "ipv4:192.0.1.1", "dst": "ipv4:192.0.1.2"},
    {"src": "ipv4:192.0.1.3", "dst": "ipv4:192.0.1.4"},
    {"src": "ipv4:192.0.1.1", "dst": "ipv4:192.0.1.4"}
  ]
}
```

4. Path-Vector Response

An extension supporting the path-vector cost-mode MUST support the following extension of Section 11.2.3.6 of [RFC7285]:

```
object {
  cost-map.DstCosts.JSONValue -> JSONString<0,*>;
  meta.cost-mode = "path-vector";
} InfoResourcePVCostMap : InfoResourceCostMap;
```

Specifically, the preceding specifies that InfoResourcePVCostMap extends InfoResourceCostMap. The body specifies that the first extension is achieved by changing the type of JSONValue defined in DstCosts of cost-map to be an array of JSONString; the second extension is that the cost-mode of meta MUST be "path-vector".

An example cost map using path-vector is the following:

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

{
    "meta" : {
      "vtag" : {
        "resource-id": "my-costmap",
        "tag": "c0ce023b8678a7b9ec00324673b98e54656d1f6d"
      }
      "cost-type" : {"cost-mode":   "path-vector",
                     "cost-metric": "bw"
      }
    }
  },

  "cost-map" : {
    "ipv4:192.0.1.1": {
              "ipv4:192.0.1.2":    ["ne57"],
              "ipv4:192.0.1.4":    ["ne57", "ne74"]
    },
    "ipv4:192.0.1.3": {
              "ipv4:192.0.1.4":    ["ne56", "ne67"]
    }
  }

  "nep-map" : {
    "ne57" : {"bw" : 100},
    "ne74" : {"bw" : 100},
    "ne56" : {"bw" : 100},
    "ne67" : {"bw" : 100}
  }
}
```

To interpret the path vectors in a cost map, an ALTO client will need access to the properties of the abstract network elements named in the path vectors. One approach is to use a network element property service (e.g., the unified properties draft [I-D.roome-alto-unified-props]). In this design, they are included as a second map in the same message, for consistency and saving of round-trips.

5. Path-Vector in IRD

Announcing the support of path vector in IRD is relatively straightforard: the mode is included as the IRD.

6. Path-Vector in Incremental Updates

A concern is whether the design allows efficient incremental updates, on either the path vectors of the element properties. For this design, the keys along the data path allow such easy encoding.

7. Security Considerations

This document has not conducted its security analysis.

8. IANA Considerations

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

9. Acknowledgments

The author thanks discussions with Xiao Shi, Xin Wang, Erran Li, Tianyuan Liu, Andreas Voellmy, Haibin Song, and Yan Luo.

10. References

10.1. 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, <http://www.rfc-editor.org/info/rfc2119>.

10.2. Informative References

[I-D.amante-i2rs-topology-use-cases]
Medved, J., Previdi, S., Lopez, V., and S. Amante, "Topology API Use Cases", draft-amante-i2rs-topology-use-cases-01 (work in progress), October 2013.

[I-D.clemm-i2rs-yang-network-topo]
Clemm, A., Medved, J., Tkacik, T., Varga, R., Bahadur, N., and H. Ananthakrishnan, "A YANG Data Model for Network Topologies", draft-clemm-i2rs-yang-network-topo-01 (work in progress), October 2014.

[I-D.lee-alto-app-net-info-exchange]
Lee, Y., Bernstein, G., Choi, T., and D. Dhody, "ALTO Extensions to Support Application and Network Resource Information Exchange for High Bandwidth Applications", draft-lee-alto-app-net-info-exchange-02 (work in progress), July 2013.

[I-D.roome-alto-unified-props]
Roome, W., "Extensible Property Maps for the ALTO Protocol", draft-roome-alto-unified-props-00 (work in progress), July 2015.

[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, <http://www.rfc-editor.org/info/rfc7285>.

Authors' Addresses

Greg Bernstein
Grotto Networking
Fremont, CA
USA

Email: gregb@grotto-networking.com

Kai Gao
Tsinghua University
Beijing Beijing
China

Email: gaok12@mails.tsinghua.edu.cn

Young Lee
Huawei
TX
USA

Email: leeyoung@huawei.com

Wendy Roome
Nokia/Bell Labs
600 Mountain Ave, Rm 3B-324
Murray Hill, NJ 07974
USA

Phone: +1-908-582-7974
Email: wendy.roome@nokia.com

Michael Scharf
Nokia
Germany

Email: michael.scharf@nokia.com

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

Email: yry@cs.yale.edu