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.

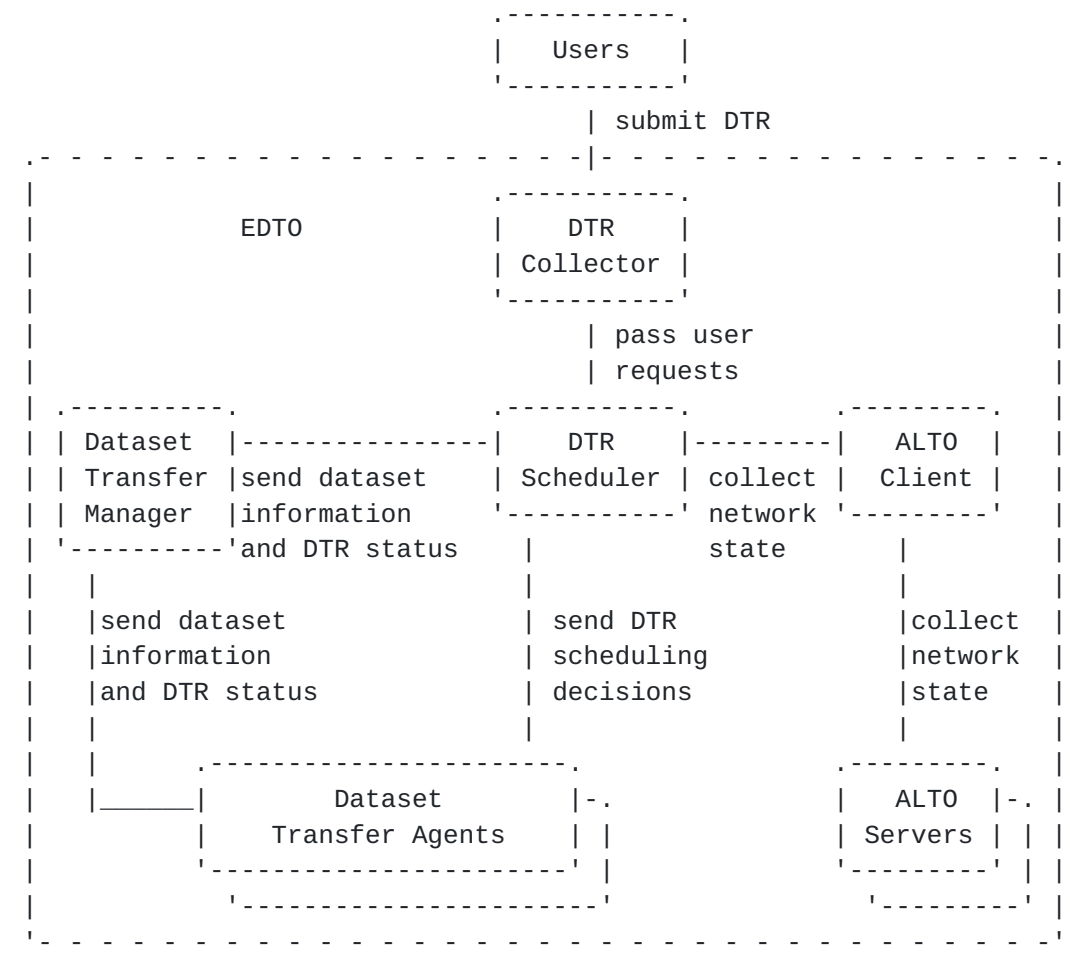
- The orchestrating framework must support the interaction between the dataset management module and the dataset transfer orchestrating module. The key information to be exchanged between modules includes dataset information, the state of the network, the flows in progress, as well as trends and network-segment 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