

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

X. Wang
S. Dong
Tongji University
G. Chen
Huawei Technologies
July 6, 2015

Design and Implementation of Large Data Transfer Coordinator
draft-wang-alto-large-data-framework-00.txt

Abstract

The Application-Layer Traffic Optimization (ALTO) protocol provides network information with the goal of improving both application performance and network resource utilization. As data transfers become larger (e.g., due to big data analysis), more data transfers are concurrent but with service requirements, and more network capabilities are emerging (e.g., SDN allowing a data transfer to request specific routes or Qos), the management of large data transfers has become an increasingly challenging issue. This document introduces Data Transfer Center (DTC), a centralized framework to coordinate and schedule large data transfers. DTC considers all three components: data transfer requirements, (ALTO) network information, and SDN control capabilities. This document specifies not only the basic framework of DTC, but also a key component, Data Transfer Set (DTS) to specify data transfers and their relations.

Status of This Memo

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

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

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

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

Copyright Notice

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

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

Table of Contents

1. Introduction	2
2. Requirements Language	3
3. Terminology and Notation	3
4. Framework	4
4.1. Architecture	4
4.2. Job Collector	5
4.3. ALTO Client	5
4.3.1. PASSIVE and ACTIVE Mode	5
4.4. Task Scheduler	5
4.4.1. Priority Model	6
4.5. DTN Controller	6
5. Data Transfer Description	7
5.1. Data Transfer Task (DTT) Description	7
5.2. Data Transfer Set (DTS) Description	8
5.3. Northbound API	8
6. Security Considerations	9
7. IANA Considerations	9
8. Acknowledgments	9
9. References	9
9.1. Normative References	9
9.2. Informative References	10
Authors' Addresses	10

1. Introduction

There is substantial need to manage large data transfers. Considering limited network resources such as bandwidth, inappropriate handling large data transfer would reduce performance significantly. It could be easier to cause network congestion than low traffic. Congested network can result in higher rate of packet loss, then triggers retransmissions, which can cripple already

heavily loaded networks. It's necessary to manage large data transfer not only for high network resource utilization but also for users' experience aspect.

Scheduling data flows needs network information such as available bandwidth between two transfer nodes. ALTO defines cost maps providing cost between two pids and endpoint cost service for two endpoints. By utilizing these network information, application can determine how to allocate bandwidth for each data flow. However, to achieve such scheduling, there needs a centralized coordinator that can be aware of every data flow requirements. Moreover, to get the customized requirements for each data transfer, a general interface is needed to obtain the correlation among data flows besides single data flow requirements.

This document introduces the design and implementation of a framework to Data Transfer Center (DTC), a centralized framework to schedule data flows based on network information and correlation among data flows. Framework design details are described in Section 4.1.

This document is organized as follows: Section 4 gives our general architecture for large data transfer. Section 5 gives details on DTS, Data Transfer Task (DTT) and northbound API under our general architecture.

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.

o DTT

Data Transfer Task. The data transfer request for a single file or directory. It's the minimum data transfer unit user can manipulate. See more detailed description in Section 5.1.

o DTS

Data Transfer Set. A group of DTTs which have relations with each other but have no relation with any other DTTs outside of the DTS it belonging to. See more detailed description in Section 5.2.

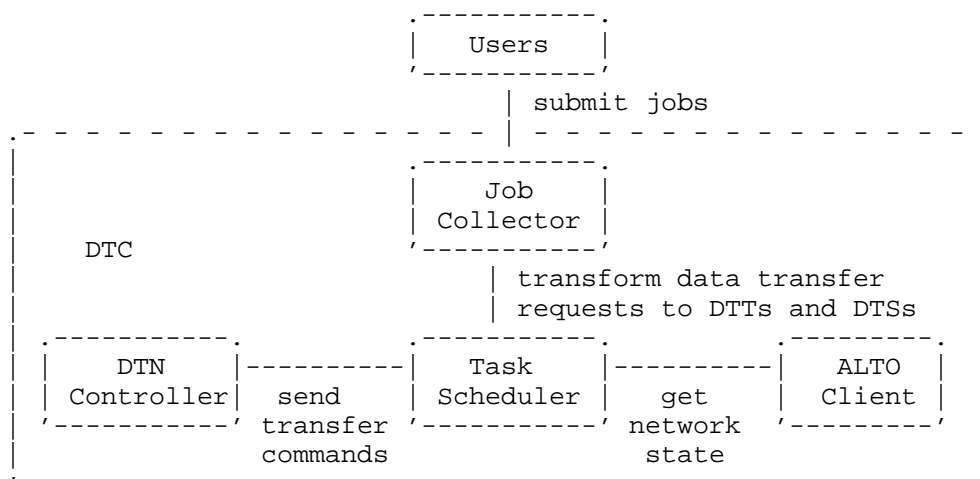
- o Relation

A relation defines how two DTTs connect. Any DTTs having relations will be grouped into the same DTS. See more detailed description in Section 5.2.

4. Framework

4.1. Architecture

This section describes the design details of four components of the framework, 1. Job Collector; 2. ALTO Client; 3. Task Scheduler; 4. DTN Controller. Among these four modules, task scheduler is the core of the framework. Job Collector provides interface to users for submitting data transfer requests, which will be transformed to DTTs and DTSSs and passed to task scheduler for further process. Task scheduler makes scheduling based on the network information generated by ALTO client as well as the requirements of each data transfer from DTS. After computing allocation of bandwidth for each DTT, task scheduler will send transfer commands to DTN controller to start data transmission. Figure 1 shows the whole process.



The benefits of DTC include:

- o 1. It can achieve better network resource (bandwidth) allocation since it manages all data transfer requirement in a centralized framework.

- o 2. It takes customized data transfer requirement into consideration by introducing DTS to capture correlation among data flows.
- o 3. It's modular to support different scheduler algorithm implementations.

4.2. Job Collector

The job collector is responsible to manage data transfer requests from user and transforming these data transfer requests to DTTs for task scheduler to process. It is important that the requests are dynamic and hence the API of the job collector allows dynamic insertion and deletion of data transfers. Details of the data transfer description and APIs for users are described in Section 5.3: Northbound API.

4.3. ALTO Client

ALTO client will be responsible to get network state to task scheduler for further usage. Although different scheduling algorithms may request different ALTO services, cost map and endpoint cost map seems to be the most useful services for scheduling tasks.

4.3.1. PASSIVE and ACTIVE Mode

ALTO client should support two modes according to the way it perceives network state changes, PASSIVE and ACTIVE. In PASSIVE mode, ALTO client will query ALTO server periodically to get latest network states. If the network state changes after one query, the ALTO client will not be aware of the change until next query. In ACTIVE mode, ALTO client will only query ALTO server once to get the initial network state. If network state changes after that, the ALTO client will be notified by ALTO server so it does not have to query ALTO server again. Note that ACTIVE mode will only be supported by ALTO server with ALTO SSE implemented.

4.4. Task Scheduler

The duty of task scheduler is to assign tasks from job collector to proper data transfer nodes (DTNs), splitting a file to several partial files to different DTNs if necessary, and notify the DTN controller to initiate the transfer. We will not discuss specific algorithm in this document but we assume algorithms used by scheduler should take network states provided by ALTO client into consideration. Different schedulers may obey different principles, some schedulers aims to maximize the number of finished tasks while some try to transfer as much data as possible.

4.4.1. Priority Model

In this section, we proposed a schedule model based on priority. In this model, every DTT will be set a predefined priority value, e.g. LOW, MEDIUM and HIGH, to indicate how important it is. The principle of this model is that DTTs with higher priority have the privilege to occupy more resources such as available bandwidth. If the priority is not set, the DTT must be set a default one. Things become tricky when user does not specify priority but an expected finish time instead. However, in this model, it is easy to be solved by transforming expected finish time to priority by following steps:

- o 01. Assign the lowest priority to the task and schedule the task.
- o 02. Calculate the task's estimated finish time. If the estimated finish time is longer than user specified finish time, increase the task priority by one and reschedule the task, else the schedule procedure completes.
- o 03. Keep doing step 2 until either the schedule procedure completes or the task is assigned as highest priority. If the task is still not able to be finished, we will keep it as highest priority and transfer as much data as possible.

The specific algorithm used to adjust the resources according to the priority is not described in this document.

4.5. DTN Controller

DTN controller is only responsible for two following functions:

- o 01. Receive and process instructions from task scheduler, e.g. starting a new transfer, aborting a running transfer and adjusting transfer parameters such as transfer rate or number of connections.
- o 02. Monitor transfer status and update status changes to task scheduler. If a transfer failed or finished, it should notify task scheduler the details for further scheduling.

If we assume task scheduler is a manager, then DTN controller are workers who focusing on its own job without caring anything else. DTN controllers are not able to communicate with each other, which means it does not have a global view. Since the DTN controller has to utilize DTNs to transfer data, it should be deployed either in a server able to access DTNs or in the DTNs themselves.

5. Data Transfer Description

Introducing a systematic description of data transfer is challenging. Although it is easy to describe each individual data transfer, this simple description method is not sufficient for a centralized data transfer coordinator because it is not capable of representing relations, e.g. dependencies, between different data transfers. To solve this problem, this section introduces the concepts Data Transfer Task (DTT) first, then it enumerates common relations between different DTTs and introduces the concept of Data Transfer Set (DTS) to represent them.

5.1. Data Transfer Task (DTT) Description

The schema for DTT representation is described as following Backus-Naur Form:

```
dtc := dtc_id src_file_candidates dst_file [requirement]
src_file_candidates := src_file {src_file_candidates}
src_file := resource_path
dst_file := resource_path
resource_path := ss_id file_path
requirement := [start_time] [finish_time] [priority]
```

with fields:

- o src_file_candidates

This field specifies the source file candidates. since there could be multiple sources for a file, the field is set as a list of source files while each source file is represented by a resource path described below.

- o dst_file

This field specifies the destination file. Unlike multiple source file candidates, there could only be one destination file for a DTT. Destination file is also represented by a resource path.

- o resource_path

This field identifies a unique resource in multiple storage systems. Since a storage system could be connected by multiple data transfer nodes, it is not accurate to identify a resource by server host and file path anymore. To solve this problem, DTC will assign every connected storage system a unique id. Thus, users can combine ss_id, which is the unique storage system id, and

file_path, which indicates location of the file in the corresponding storage system, to identify a unique resource.

- o requirements

This field specifies the requirement of the task. Currently, the DTT defines three kinds of requirements, start_time, finish_time and priority.

5.2. Data Transfer Set (DTS) Description

The schema for DTS representation is described as following Backus-Naur Form:

```

dts := dts_id relation_list
relation_list := relation {relation_list}
relation := depend | sharing
depend := dtt_id "depend" dtt_id
share := dtt_id "share" dtt_id requirement
requirement := "start_time" | "finish_time"
```

- o relation

This field indicates how two DTTs connect. This document specifies two kinds of relations, "depend" and "share". "depend" relation points out that one DTT depends on another DTT, which means the former DTT must not start until the latter completes. "share" relation points out that two DTTs share the same requirement. For example, if two DTTs share the same finish_time, they are expected to finish at the same time.

"depend" and "share" relation are both transitive. If DTT_1 depends on DTT_2, and DTT_2 depends on DTT_3, then DTT_1 depends on DTT_3. If DTT_4 shares requirement R with DTT_5, and DTT_5 shares requirements R with DTT_6, then DTT_4 shares requirements R with DTT_6.

5.3. Northbound API

Normally, users will submit a group of DTTs at the same time to submit a DTS. While DTS is running, the user should be able to add DTTs to or remove DTTs from DTS dynamically. To enable these features, a job collector should provide the following five functions for user:

- o submitDTS(dtt_list)

This function creates a new DTS. It accepts a list DTT as parameter and must return a DTS id for user to identify the DTS created. If the creation fails, it must throw an error.

- o abortDTS(dts_id)

This function aborts a running DTS. It accepts a dts_id parameter and must abort all DTTs belonging to the DTS. The function return value should indicate if the abort action succeeds or not. If the DTS does not exist, it must throw an error.

- o addDTT2DTS(dts_id, dtt)

This function adds a new DTT to a existing DTS. This function accepts a dtt_id and a list of DTS as parameters. The function return value should indicate if the add action succeeds or not.

- o removeDTTFromDTS(dtt_id, src_resource, dst_resource)

This function removes a DTT from a existing DTS. This function accepts a dtt_id and a source and destination resource path pair. The source and destination resource pair will identify a unique DTT to be removed. The function return value should indicate if the remove action succeeds or not.

6. Security Considerations

This document has not conducted its security analysis.

7. IANA Considerations

This document does not specified its IANA considerations, yet.

8. Acknowledgments

The authors thank discussions with Yicheng Qian.

9. References

9.1. Normative References

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

9.2. Informative References

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

Authors' Addresses

Xin Wang
Tongji University
4800 Cao'an Road, Jiading District
Shanghai
China

Email: xinwang2014@hotmail.com

Shu Dong
Tongji University
4800 Cao'an Road, Jiading District
Shanghai
China

Email: dongs2011@gmail.com

Guohai Chen
Huawei Technologies
101 Software Avenue, Yuhua District
Nanjing
China

Email: chenguohai@huawei.com