Workgroup:
Application-Layer Traffic Optimization
Internet-Draft:
draft-giraltyellamraju-alto-bsg-multidomain-00
Published: 24 October 2022
Intended Status: Informational
Expires: 27 April 2023
Authors: J. Ros-Giralt — Qualcomm Europe, Inc.
S. Yellamraju — Qualcomm Technologies, Inc.
Q. Wu — Huawei
L. M. Contreras — Telefonica
R. Yang — Yale University
K. Gao — Sichuan University

Bottleneck Structure Graphs in Multidomain Networks: Introduction and Requirements for ALTO

Abstract

This document proposes an extension to the base Application-Layer Traffic Optimization(ALTO) protocol to support the computation of bottleneck structure graphs [I-D.draft-giraltyellamraju-alto-bsg-requirements] under partial information. A primary application corresponds to the case of multi-domain networks, whereby each network domain is administered separately and lacks information about the other domains. A proposed border protocol is introduced that ensures each domain's independent convergence to the correct bottleneck substructure graph without the need to know private flow and topology information from other domains. Initial discussions are presented on the necessary requirements to integrate the proposed capability into the ALTO standard.

About This Document

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

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

Discussion of this document takes place on the Application-Layer Traffic Optimization Working Group mailing list (mailto:alto@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/alto/. Subscribe at https://www.ietf.org/mailman/listinfo/alto/.

Source for this draft and an issue tracker can be found at https://github.com/giralt/draft-ietf-alto-gradientgraph-multidomain.

Status of This Memo

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

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

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

This Internet-Draft will expire on 27 April 2023.

Copyright Notice

Table of Contents

1. Introduction

Bottleneck structures have been recently introduced in [G2-SIGCOMM] and [G2-SIGMETRICS] as efficient computational graphs that embed information about the topology, routing and flow information of a network. These computational graphs allow network operators and application service providers to compute network derivatives that can be used to make traffic optimization decisions. For instance, using the bottleneck structure of a network, a real-time communication (RTC) application can efficiently estimate the multi-hop end-to-end available bandwidth, and use that information to tune the encoder's transmission rate and optimize the user's Quality of Experience (QoE). Bottleneck structures can be used by the application to address a wide variety of communication optimization problems, including routing, flow control, flow scheduling, bandwidth prediction, and network slicing, among others.

The ALTO draft [I-D.draft-giraltyellamraju-alto-bsg-requirements] introduces a new abstraction called Bottleneck Structure Graph (BSG) and the necessary initial requirements to integrate it into the existing ALTO services (Network Map, Cost Map, Entity Property Map and Endpoint Cost Map) exposing the properties of the bottleneck structure to help optimize application performance. When the ALTO server has full visibility of the network (i.e., all of its links, routes, and flows), the bottleneck structure can be computed using the algorithm introduced in [G2-SIGCOMM] [G2-SIGMETRICS]. In many scenarios, however, flows traverse multiple autonomous systems (ASs), and thus an ALTO server deployed in one AS may not have access to topological and flow information from the other domains. In this document, we describe a border protocol that allows ALTO servers in each AS to share their local path metrics dictionary (obtained via their local computation of the bottleneck structure graph) with their neighbouring ASs. Using the algorithm introduced in this document, this information alone is enough to ensure independent convergence by each AS to the correct bottleneck structure. This cooperative solution presents similar properties as those found in well-known IETF protocols such as BGP, including the properties of scalability (since metrics only need to be shared on a per-path rather than per-flow basis, and only between neighboring ASs) and privacy (since no sensitive flow or topology information needs to be shared).

We also present initial discussions on the necessary requirements to integrate the proposed capability into the ALTO standard.

2. Conventions and Definitions

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

3. Distributed Protocol to Compute the Bottleneck Structure of an AS

3.1. Motivation

In many real-world communication problems, data flows need to traverse multiple network domains, each one administered by a different operator that is responsible for (1) maintaining its own (and only its own) domain and (2) ensuring interoperability with the other domains. The quintessential example of multi-domain networks is the Internet, designed as a "network of interconnected networks", commonly known as *autonomous systems* (ASs).

In multi-domain networking environments, the operator in each domain only has visibility of its own network. In particular, the operator may know with precision the topology, the capacity of each link, the classes of quality of service (QoS) to serve, and the flows currently active in their network, but usually has no knowledge about the structure and state of any other network in the multi-domain environment. For instance, a data flow may need to cross two network domains, one operated by Operator O1 and another one operated by Operator O2. O1 has no visibility into the network operated by O2, while O2 has no visibility into the network operated by O1. Yet both networks need to cooperate in order to ensure the end-to-end QoS required by the flow.

Bottleneck structures ([G2-SIGCOMM], [G2-SIGMETRICS]) are computational graphs that characterize the state of a communication network allowing human operators and machines to compute network derivatives very fast. These derivatives are core building blocks that enable the optimization of networks in a variety of problems including traffic engineering, routing, flow scheduling, capacity planning, resilience analysis and network design. In order to compute the bottleneck structure of a network, information of the set of links traversed by each flow and the capacity of the links is required. In a multi-domain networking environment, however, such information is only known partially. For instance, in the example above, operator O1 can know the set of links traversed by a flow that reside in its own network, but may not know the set of links

traversed by a flow that reside in the operator O2 network. Moreover, in many cases, such information is considered confidential for security, privacy and competitiveness reasons.

In this document, we introduce a distributed protocol that addresses the above problem, enabling the computation of bottleneck structures under the scenario of partial information. In particular, an algorithm to compute the bottleneck structure of a network domain--referred as the *bottleneck substructure*--is introduced that only requires a simple, scalable, and secure cooperative exchange of a path metric between neighboring autonomous systems to ensure global convergence to the correct state. Because network operators do have the ability to cooperate (provided that the exchange is simple, secure and guarantees privacy), the algorithm provides a practical methodology to optimize system-wide flow performance in a multi-domain network.

3.2. Base Protocol Definitions

In this section, we briefly state the basic definition of bottleneck structure and introduce a few simple additional definitions that are necessary to understand the proposed protocol. For a more thorough description of the bottleneck structure framework, please refer to [I-D.draft-giraltyellamraju-alto-bsg-requirements].

Let L and F be the set of links and flows of a network, respectively. Its bottleneck structure is defined as follows:

- Links and flows are represented by vertices in the graph.
- There is a directed edge from a link l to a flow f if and only if flow f is bottlenecked at link l.
- There is a directed edge from a flow f to a link l if and only if flow f traverses link l.

Since the above definition corresponds to a graph, we use the terms *bottleneck structure* and *bottleneck structure graph* (BSG) interchangeably.

As shown in [I-D.draft-giraltyellamraju-alto-bsg-requirements], the BSG explains how perturbations in a network (e.g., the arrival or departure of a flow, the change in link capacity of a network, a link failure, etc.) propagate through the network. Mathematically, these perturbations can be understood as network derivatives. Because these derivates can be computed in the graph as simple delta calculations, the BSG enables a computationally scalable mechanism to optimize a network for a variety of use cases such as optimal path computation, bandwidth prediction, service placement, or

network topology reconfiguration, among others ([G2-SIGCOMM], [G2-SIGMETRICS]).

To achieve scalability, the protocol proposed in this document uses a version of the bottleneck structure graph called *Path Gradient Graph* (PGG) (see [I-D.draft-giraltyellamraju-alto-bsg-requirements]). The PGG significantly reduces the size of the bottleneck structure graph by collapsing all the vertices of the flows that follow the same path into a single vertex called the *path vertex*. This technique leads to a more compact representation of the bottleneck structure graph (thus, significantly reducing computational complexity and memory storage) without affecting its accuracy.

The following table introduces additional conventions and definitions that are used in the description of the distributed protocol in the next section:

Notation	Description
A	The set of autonomous systems (ASs).
A_i	An AS in A, for i = 1, ..., |A|.
P(A_i) = {p_1, ..., p_|P(A_i)|}	The set of active paths found in A_i. These are paths for which there exist 0traffic flowing through them.
L(A_i) = {l_1, ..., l_|L(A_i)|}	The set of active links found in A_i. These are links for which there exists traffic flowing through them.
B	The global bottleneck structure graph. The form of bottleneck structure used by the distributed algorithm introduced in this document is the Path Gradient Graph [I-D.draft-giraltyellamraju-alto-bsg-requirements].
B.BW(p)	The bandwidth available to path p according to the global bottleneck structure. This is always the globally correct available bandwidth for path p.
B(A_i)	The bottleneck substructure of A_i, corresponding to the subgraph of B that includes (1) the vertices corresponding to the paths in P(A_i), (2) the vertices corresponding to the links in L(A_i) and (3) all the edges in B that connect them. If a path p in P(A_i) is bottlenecked at a link not in L(A_i), then B(P, L) includes a virtual link v with capacity equal to B.BW(p) and a directed edge from v to p.
B(A_i).BW(p)	The bandwidth available to path p according to the bottleneck substructure of A_i. This value is equal to B.BW(p) when the distributed algorithm terminates.
PL(A_i)	A dictionary mapping every path in P(A_i) with the subset of links in L(A_i) that it traverses. Note that a path p can traverse one or more links not in L(A_i).

Notation	Description
	This reflects the notion of partial information inherent to multi-domain networking environments. That is, A_i may not know all the links traversed by its active paths; in particular, it only knows the subset of links that are in A_i.
C(A_i)	A dictionary mapping each link in A_i with its capacity (in bps).
N(A_i)	The set of ASs that are neighbors of A_i.
PM(A_i)(p)	The current bandwidth available to path p as computed by A_i. This is also known as the path metric of p according to A_i.

Table 1: Conventions and definitions used in the description of the distributed protocol.

3.3. Description of The Distributed Protocol

The algorithm run by each autonomous system AS_i, 1 <= i <= |A|, consists of two independently executed events as follows:

Event: TIMER

- Every s seconds, perform the following tasks:

    1. B(A_i) = COMPUTE_BOTTLENECK_SUBSTRUCTURE(L(A_i), PL(A_i), C(A

    2. PM(A_i)(p) = B(A_i).BW(p), for all p in P(A_i);

    3. For all A_j in N(A_i):

        3.1 Send to A_j a PATH_METRIC_ANNOUNCEMENT message including

Event: PATH_METRIC_EXCHANGE

- Upon receiving a PATH_METRIC_ANNOUNCEMENT from AS_j carrying (AS_j

    1. PM(A_i)(p) = min{PM(A_i)(p), PM(A_j)(p)}, for all p in P(A_i)

As shown above, using a PATH_METRIC_ANNOUNCEMENT message, each AS periodically shares local path metric information with its neighbor ASs. It can be shown that this information alone is enough to ensure the convergence of all the participating ASs to their correct bottleneck substructure. (This is similar to the way BGP [RFC4271] sends *Update Messages* to converge to a globally correct routing table by only exchanging local knowledge between neighbor ASs.)

The procedure COMPUTE_BOTTLENECK_SUBSTRUCTURE is called from the TIMER event, and it is responsible for computing the bottleneck substructure. It can be proven that this procedure converges to the

correct bottleneck substructure within a finite number of PATH_METRIC_ANNOUNCEMENT messages:

Procedure: COMPUTE_BOTTLENECK_SUBSTRUCTURE(L, PL, C, PM):

```
 1. i = 0; L_0 = L; PL_0 = PL;

 2. While True:

        2.1. B_i = COMPUTE_BOTTLENECK_STRUCTURE(L_i, PL_i, C);

        2.2. If B_i.BW(p) == PM(p) for all path p in PL_i:

                2.2.1. Break;

        2.3. For all path p in PL_i such that B_i.BW(p) > PM(p):

                2.3.1. If PL_i[p] has no virtual link:

                        2.3.1.1. Add a new virtual link v to the set

                        2.3.1.2. Add virtual link v to the set L_i;

                2.3.2.  Set C(v) = PM(p);

        2.2. i = i + 1;

        2.5. L_i = L_{i-1};

        2.6. PL_i = PL_{i-1};

 3. Return B_i;
```

In the above procedure, the function COMPUTE_BOTTLENECK_STRUCTURE corresponds to the GradientGraph algorithm introduced in [G2-TREP].

The termination condition of this procedure is found in line 2.2.1:

```
 B_i.BW(p) == PM(p) for all path p in PL_i
```

When the distributed algorithm converges to a final solution, the invocation of the procedure COMPUTE_BOTTLENECK_SUBSTRUCTURE returns immediately at this condition, and the path metric dictionaries for all the autonomous systems (PM(A_i) for 1 <= i <= |A|) no longer change, provided that the network state does not change. Further, upon termination, the distributed algorithm ensures that all the path metric values for all the autonomous systems are in agreement:

```
 PM(A_i)(p) == PM(A_j)(p) for all p in A_i, p in A_j, A_i in A and A_
```

We call this the *convergence condition*, to denote the fact that upon termination, all the path metrics from all the ASs are in agreement.

3.4. Example: Global Convergence to the Correct Bottleneck Substructures

Figure 1 shows an example of a multidomain network with two autonomous systems, AS1 (the upper subdomain) and AS2 (the lower subdomain). Each link li is represented by a squared box and has a capacity ci. For instance, link l1 is represented by the top most squared box and has a capacity of c1=25 units of bandwidth. In addition, each path is represented by a line that passes through the set of links it traverses. For instance, path p6 traverses links l1, l2 and l3.

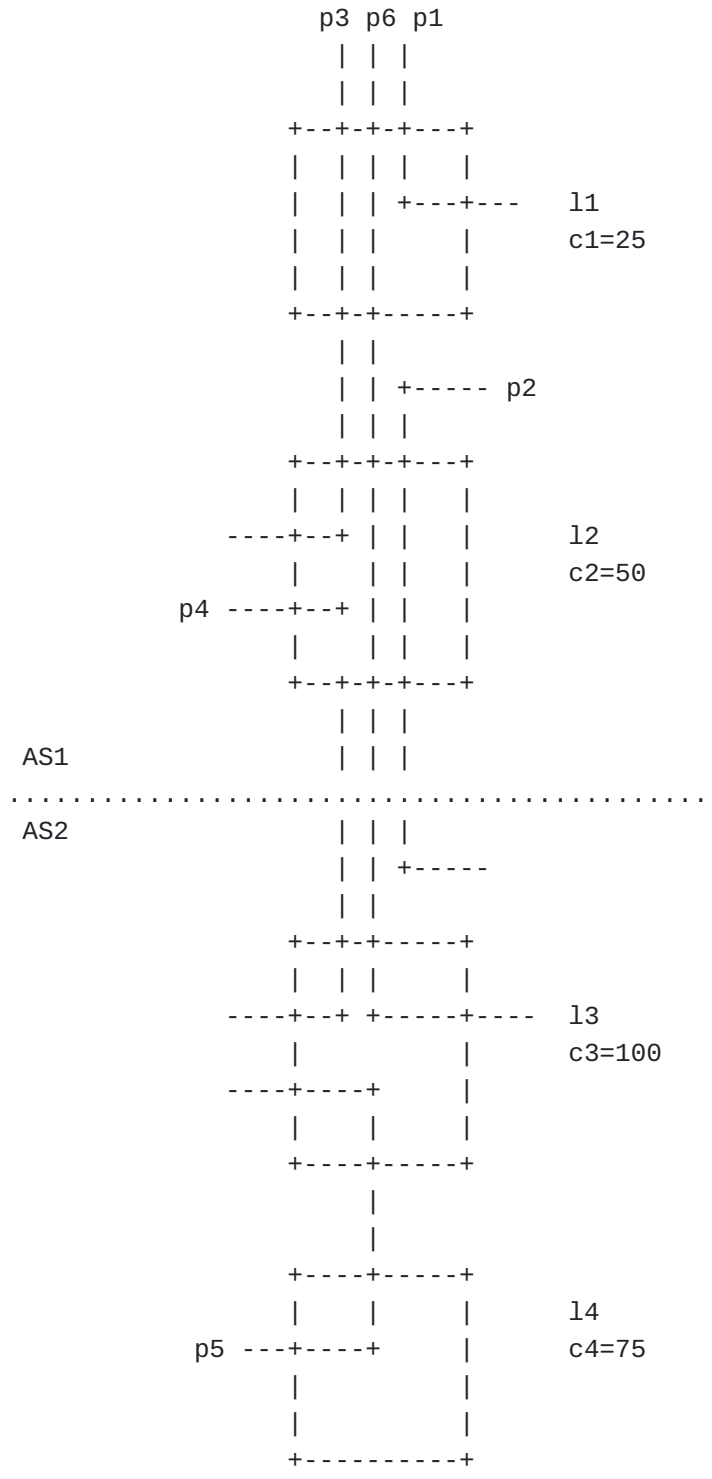


Figure 1: Multi-domain network example with two autonomous systems.

The global bottleneck structure of this network corresponds to the following digraph (see [I-D.draft-giraltyellamraju-alto-bsg-requirements] for details on how a bottleneck structure is computed):

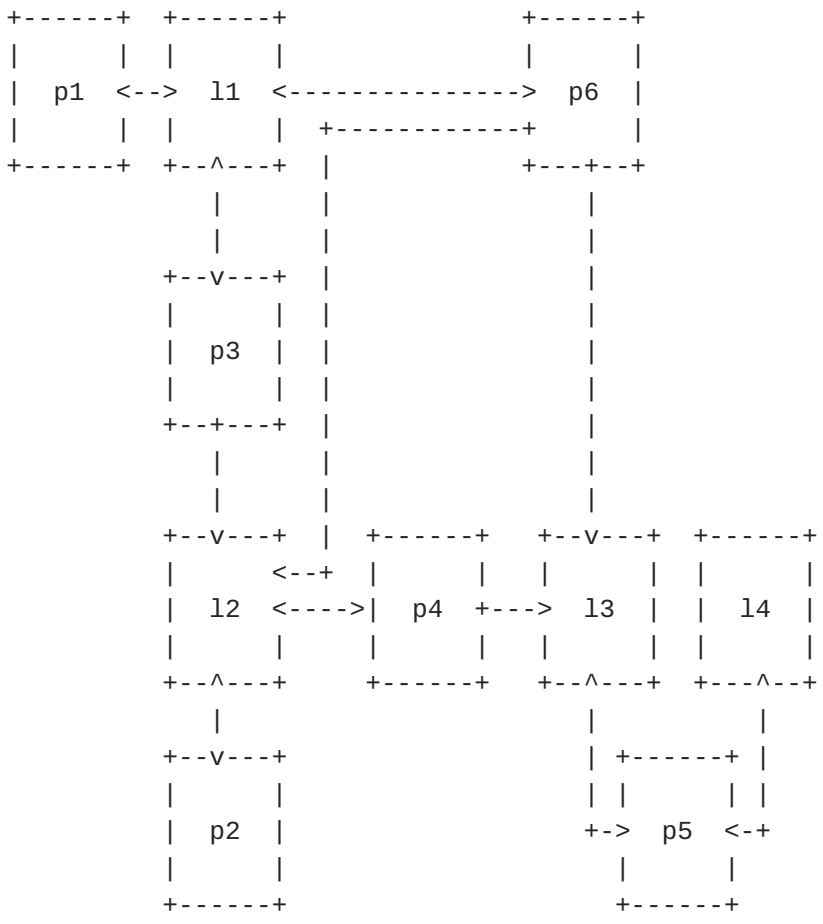


Figure 2: Global bottleneck structure of the network in Figure 1.

Using the definitions introduced in Table 1, we have that the bottleneck substructure for AS1 and AS2 (that is, B(AS1) and B(AS2)) are as shown in Figure 3 and Figure 4, respectively.

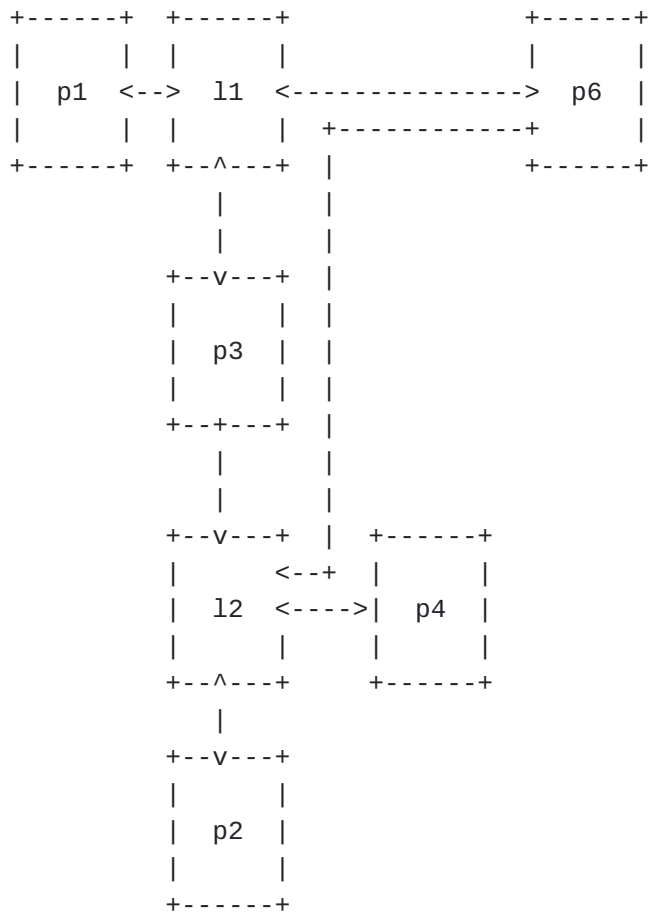


Figure 3: B(AS1): Bottleneck substructure of AS1.

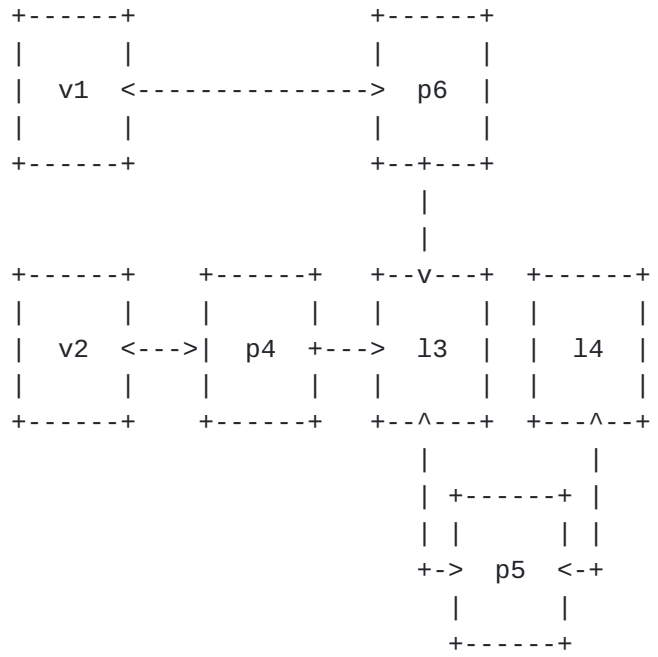


Figure 4: B(AS2): Bottleneck substructure of AS2.

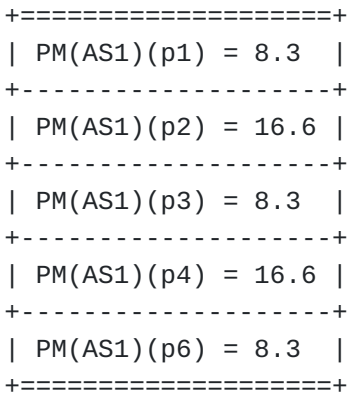
Note that according to the definition in Table 1, the bottleneck substructure of each AS corresponds to the subgraph of the global bottleneck structure B that includes all the vertices and edges in B that correspond to paths and vertices observed in the AS, plus an additional virtual link for each path that is bottlenecked outside the AS. In particular, AS2 has two virtual links v1 and v2 associated with paths p6 and p4, respectively, since these two paths are bottlenecked outside AS2. Similarly, AS1 has no virtual links because all of its paths are bottlenecked in its own domain.

The objective consists in computing the bottleneck substructure of all the ASs in a distributed manner when each AS only has local information about the state of its links and paths. The distributed protocol introduced in this document provides a solution to this problem.

Figure 5 and Figure 6 present the step-by-step execution of the distributed protocol as run by AS1 and AS2, respectively. For each iteration of the protocol, the state of the local path metric dictionary PM(AS) and of the bottleneck substructure B(AS) are presented.

Iteration 1:

State of the path metric dictionary PM(AS1):



State of the bottleneck substructure B(AS1):

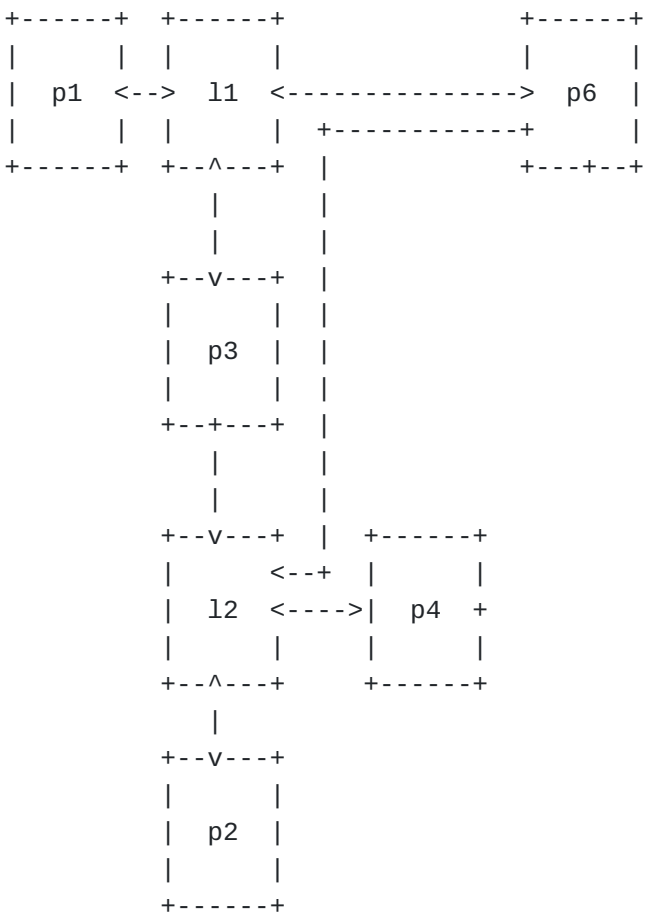
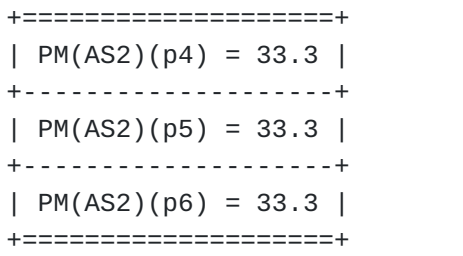


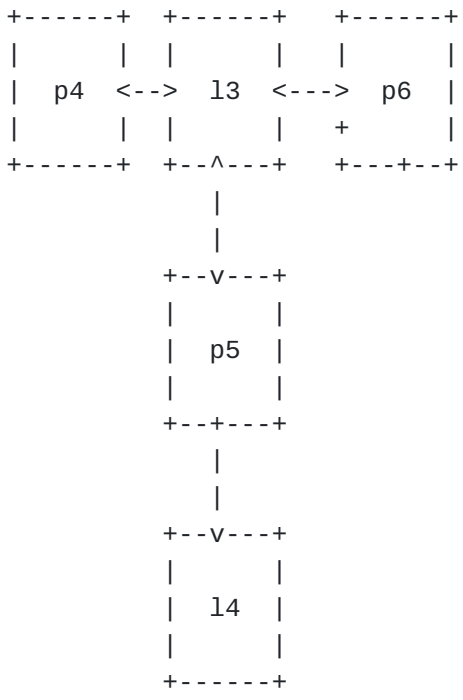
Figure 5: Execution of the distributed protocol by AS1.

Iteration 1:

State of the path metric dictionary PM(AS2):

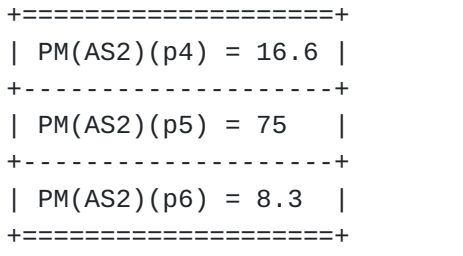


State of the bottleneck substructure B(AS2):



Iteration 2:

State of the path metric dictionary PM(AS2):



State of the bottleneck substructure B(AS2):

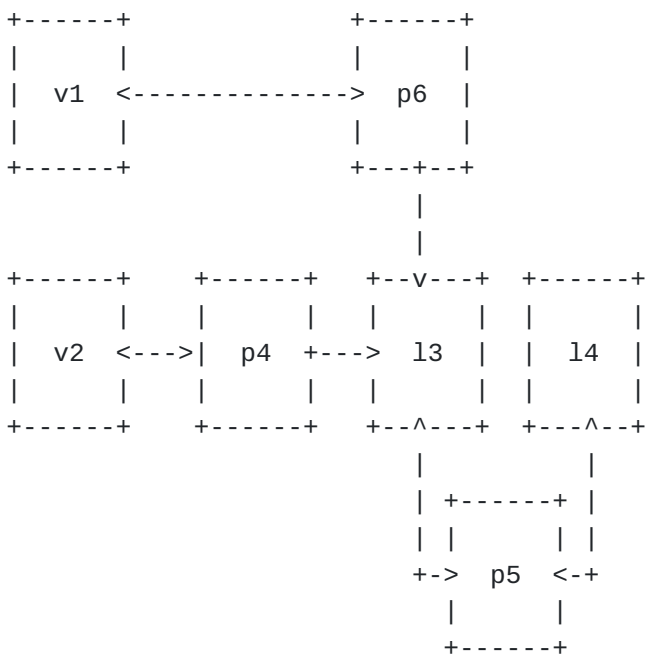

v1
p6
v2
p4
l3
l4
p5

Figure 6: Execution of the distributed protocol by AS2.

Note that at the end of the execution of the distributed algorithm, the convergence condition

PM(A_i)(p) = PM(A_j)(p) for all p in A_i, p in A_j, A_i in A and A_j

is satisfied, as shown in Table 2.

p	PM(A1)(p)	PM(A2)(p)
p1	8.3	--
p2	16.6	--
p3	8.3	--
p4	16.6	16.6
p5	--	75
p6	8.3	8.3

Table 2: Verification of the convergence condition.

4. Requirements

This section provides a discussion on the necessary requirements to integrate the proposed distributed protocol into the ALTO standard. Throughout this discussion, we assume without loss of generality that each AS is managed by an ALTO server, and that each server only has visibility of the topology, links and flow state of the AS it is managing. We also assume that the TIMER and the PATH_METRIC_EXCHANGE events are executed by each ALTO server. An alternative architecture could consider executing these events in a separated engine, and have the ALTO server query this engine to obtain the final bottleneck structures, decoupling the distributed protocol from the ALTO standard. While this approach might be desirable in some cases, we currently omit it from this discussion since it is relatively simpler from an integration requirements standpoint.

To implement the proposed distributed protocol using ALTO, two broad requirements are necessary:

* Requirement 1: The capability for each ALTO server to compute bottleneck substructures of its own AS.

* Requirement 2: The capability for each ALTO server to communicate with its neighboring ASs.

4.1. Requirement 1: Computation of Bottleneck Substructures

The requirements for an ALTO server to compute the bottleneck substructure of its associated AS are the same as the requirements

to compute the bottleneck structure in the case the network consists of a single autonomous system. These requirements are discussed in the Requirements Section of [I-D.draft-giraltyellamraju-alto-bsg-requirements]. Refer to this document for further details.

4.2. Requirement 2: Communication Between Neighboring ASs

The TIMER event executed by each ALTO server needs to periodically transmit a PATH_METRIC_ANNOUNCEMENT message to its neighboring ASs. This leads to the following requirement:

- Requirement 2.1: ALTO servers managing neighboring ASs need to be reachable to each other.

- Requirement 2.2: The sharing of algorithmic state between ALTO servers requires extending the base ALTO protocol to support server-to-server communication semantics.

This requirement constitutes a new capability, since the current ALTO standard only supports client-to-server communication semantics [RFC7285].

We note that [I-D.draft-zhang-alto-oam-yang] discusses mechanisms for cross-ALTO server communication with the objective to facilitate Operations and Management (OAM) of multi-server deployments. The distributed protocol proposed in this document could be used as a use case to help drive the specifications of the inter-server communication protocol discussed in [I-D.draft-zhang-alto-oam-yang] or in any future ALTO RFCs that may focus on sharing of algorithmic state.

5. Security Considerations

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

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

For confidentiality of ALTO information, a network operator should be aware that this extension may introduce a new risk: As the Bottleneck Structure information may reveal more fine-grained internal network structures than the base protocol, an attacker may

identify the bottleneck link and start a distributed denial-of-service (DDoS) attack involving minimal flows to conduct in-network congestion. Given the potential risk of leaking sensitive information, the BSG extension is mainly applicable in scenarios where:

- The properties of the Bottleneck Structure Graph do not impose security risks to the ALTO service provider; e.g., by not carrying sensitive information.

- The ALTO server and client have established a reliable trust relationship, for example, operated in the same administrative domain, or managed by business partners with legal contracts and proper authentication and privacy protocols.

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

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

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

For availability of ALTO service, an ALTO server should be cognizant that using Bottleneck Structures might have a new risk: frequently querying the BSG service might consume intolerable amounts of computation and storage on the server side. For example, if an ALTO server implementation dynamically computes the Bottleneck Structure for each request, the BSG service may become an entry point for denial-of-service attacks on the availability of an ALTO server. To mitigate this risk, an ALTO server may consider using optimizations such as precomputation-and-projection mechanisms [MERCATOR] to reduce the overhead for processing each query. An ALTO server may also protect itself from malicious clients by monitoring the

behaviors of clients and stopping serving clients with suspicious behaviors (e.g., sending requests at a high frequency).

6. IANA Considerations

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

7. References

7.1. Normative References

[I-D.draft-giraltyellamraju-alto-bsg-requirements] Ros-Giralt, J., Yellamraju, S., Wu, Q., Contreras, L. M., Yang, Y. R., and K. Gao, "Supporting Bottleneck Structure Graphs in ALTO: Use Cases and Requirements", Work in Progress, Internet-Draft, draft-giraltyellamraju-alto-bsg-requirements-03, 23 September 2022, <https://datatracker.ietf.org/doc/html/draft-giraltyellamraju-alto-bsg-requirements-03>.

[I-D.draft-zhang-alto-oam-yang] Zhang, J., Dhody, D., Gao, K., and R. Schott, "A Yang Data Model for OAM and Management of ALTO Protocol", Work in Progress, Internet-Draft, draft-zhang-alto-oam-yang-02, 7 March 2022, <https://datatracker.ietf.org/doc/html/draft-zhang-alto-oam-yang-02>.

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

[RFC4271] Rekhter, Y., Ed., Li, T., Ed., and S. Hares, Ed., "A Border Gateway Protocol 4 (BGP-4)", RFC 4271, DOI 10.17487/RFC4271, January 2006, <https://www.rfc-editor.org/rfc/rfc4271>.

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

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

7.2. Informative References

[G2-SIGCOMM]
Ros-Giralt, J., Amsel, N., Yellamraju, S., Ezick, J., Lethin, R., Jiang, Y., Feng, A., Tassiulas, L., Wu, Z., and K. Bergman, "Designing data center networks using bottleneck structures", ACM SIGCOMM , 2021.

[G2-SIGMETRICS]
Ros-Giralt, J., Bohara, A., Yellamraju, S., Langston, H., Lethin, R., Jiang, Y., Tassiulas, L., Li, J., Tan, Y., and M. Veeraraghavan, "On the Bottleneck Structure of Congestion-Controlled Networks", ACM SIGMETRICS , 2020.

[G2-TREP]
Ros-Giralt, J., Amsel, N., Yellamraju, S., Ezick, J., Lethin, R., Jiang, Y., Feng, A., Tassiulas, L., Wu, Z., and K. Bergman, "A Quantitative Theory of Bottleneck Structures for Data Networks", Qualcomm Technologies Inc., Technical Report , 2021.

[MERCATOR] Xiang, Q., Zhang, J., Wang, X., Guok, C., Le, F., MacAuley, J., Newman, H., and Y. Yang, "Toward Fine-Grained, Privacy-Preserving, Efficient Multi-Domain Network Resource Discovery", IEEE/ACM IEEE/ACM IEEE Journal on Selected Areas of Communication 37(8): 1924-1940, 2019, <https://doi.org/10.1109/JSAC.2019.2927073>.

[NOVA] Gao, K., Xiang, Q., Wang, X., Yang, Y., and J. Bi, "An objective-driven on-demand network abstraction for adaptive applications", IEEE/ACM IEEE/ACM Transactions on Networking (TON) Vol 27, no. 2 (2019): 805-818., 2019, <https://doi.org/10.1109/IWQoS.2017.7969117>.

Authors' Addresses

Jordi Ros-Giralt
Qualcomm Europe, Inc.

Email: jros@qti.qualcomm.com

Sruthi Yellamraju
Qualcomm Technologies, Inc.

Email: yellamra@qti.qualcomm.com

Qin Wu
Huawei

Email: bill.wu@huawei.com

Luis Miguel Contreras
Telefonica

Email: luismiguel.contrerasmurillo@telefonica.com

Richard Yang
Yale University

Email: yry@cs.yale.edu

Kai Gao
Sichuan University

Email: kaigao@scu.edu.cn