UNIVERSIDADE ESTADUAL DE CAMPINAS
Faculdade de Engenharia Elétrica e de Computação

Danny Alex Lachos Peréz

Delivering Application-Layer Traffic Optimization services based on public routing data at Internet eXchange Points

Serviços para otimização do tráfego de aplicações a partir de informações públicas de roteamento nos Pontos de Troca de Tráfego na Internet

CAMPINAS

2016

Danny Alex Lachos Peréz

Delivering Application-Layer Traffic Optimization services based on public routing data at Internet eXchange Points

Serviços para otimização do tráfego de aplicações a partir de informações públicas de roteamento nos Pontos de Troca de Tráfego na Internet

Dissertation presented to the Faculty of Electrical and Computer Engineering of the University of Campinas in partial fulfillment of the requirements for the degree of Master in Electrical Engineering, in the area of Computer Engineering.

Dissertação apresentada à Faculdade de Engenharia Elétrica e Computação da Universidade Estadual de Campinas como parte dos requisitos exigidos para a obtenção do título de Mestre em Engenharia Eletrica, na Àrea de Engenharia de Computação.

Supervisor: Prof. Dr. Christian Rodolfo Esteve Rothenberg

Este exemplar corresponde à versão final da dissertação defendida pelo aluno Danny Alex Lachos Peréz, e orientada pelo Prof. Dr. Christian Rodolfo Esteve Rothenberg

CAMPINAS

2016

Agência(s) de fomento e nº(s) de processo(s): FUNCAMP

Ficha catalográfica
Universidade Estadual de Campinas
Biblioteca da Área de Engenharia e Arquitetura
Rose Meire da Silva - CRB 8/5974

L118d

Lachos Peréz, Danny Alex, 1983-
Delivering application-layer traffic optimization services based on public routing data at Internet eXchange points / Danny Alex Lachos Peréz. – Campinas, SP : [s.n.], 2016.

Orientador: Christian Rodolfo Esteve Rothenberg.
Dissertação (mestrado) – Universidade Estadual de Campinas, Faculdade de Engenharia Elétrica e de Computação.

1. Roteamento (Administração de redes de computadores). 2. Engenharia de tráfego. 3. Redes de computadores. 4. Banco de dados. I. Esteve Rothenberg, Christian Rodolfo,1982-. II. Universidade Estadual de Campinas. Faculdade de Engenharia Elétrica e de Computação. III. Título.

Informações para Biblioteca Digital

Título em outro idioma: Serviços para otimização do tráfego de aplicações a partir de informações públicas de roteamento nos Pontos de troca de tráfego na internet
Palavras-chave em inglês:
Routing (Computer network management)
Traffic engineering
Computer networks
Database
Área de concentração: Engenharia de Computação
Titulação: Mestre em Engenharia Elétrica
Banca examinadora:
Christian Rodolfo Esteve Rothenberg [Orientador]
Marcos Antonio de Siqueira
Edmundo Roberto Mauro Madeira
Data de defesa: 08-07-2016
Programa de Pós-Graduação: Engenharia Elétrica

COMISSÃO JULGADORA - DISSERTAÇÃO DE MESTRADO

Candidato: Danny Alex Lachos Peréz RA: 153840
Data da Defesa: 08 de Julho de 2016
Título da Tese:
"Delivering Application-Layer Traffic Optimization services based on public routing data at Internet eXchange Points"
"Serviços para otimização do tráfego de aplicações a partir de informações públicas de roteamento nos Pontos de Troca de Tráfego na Internet"

Prof. Dr. Christian Rodolfo Esteve Rothenberg (Presidente, FEEC/UNICAMP)
Prof. Dr. Edmundo Roberto Mauro Madeira (IC/UNICAMP) - Membro Titular
Prof. Dr. Marcos Antonio de Siqueira (PADTEC) - Membro Titular

Ata de defesa, com as respectivas assinaturas dos membros da Comissão Julgadora, encontra-se no processo de vida acadêmica do aluno.

I dedicate this master dissertation to my wife Evelin and my son Alexander for their patience, understanding and unconditional support. Without them this work would not have been possible.

I also dedicate this work to my mother Mrs. Hilda Perez, who, with her example and dedication, knew how to instill humility and the desire for self-improvement in me.

Last, but certainly not least, I dedicate this work to my siblings Veronica, Rossio and Daniel for all their support in spite of the physical distance that sometimes separates us. I also dedicate it to my little kids Aidric, Zoe and Aeowyn.

Acknowledgements

First of all, I would like to thank my advisor Prof. Dr. Christian Rothenberg for the trust and for letting me be part of his selected group of students. I would not be able to imagine the undertaking of this research without his innovative ideas, consistent support and continuous encouragement. I would therefore like to express my gratitude for being a great instructor and an even greater friend.

At the State University of Campinas (Unicamp) I have had the opportunity to learn from the best professors. I would like to acknowledge Prof. Dr. Mauricio Magalhães, Prof. Dr. Edson Borin, Prof. Dr. Edmundo Madeira and Prof. Dr. Léo Pini for sharing their knowledge and experience, which served as a constant motivation throughout this work.

I would like to thank the committee members Marcos Antonio de Siqueira, Edmundo Roberto Mauro Madeira and Christian Rothenberg for the recommendations and insightful criticism they offered me in order to improve the final version of my thesis.

I had the good fortune to meet very talented people within the INTRIG and LCA groups. My special thanks go to Mateus, Samuel, Samira, Gyanesh, Alex, Luis, Raphael, Anderson, Hirley, Ramon, Alaelson, Claudio, Talita, Javier, Elias, Rodrigo, Roberto, Mariana, Ranyeri, Aldo, Wallace, Vitor, Raphael V. and Amadeu, Mariana and Lino for their friendship and their support throughout these two years.

I would like to express special gratitude for the financial and technical support received from the Ericsson Innovation Center (Brazil), which allowed me to carry out this research.

“Discipline, sooner or later, will defeat intelligence.”
“A disciplina cedo ou tarde vencerá a inteligência.”
“La disciplina tarde o temprano vencerá a la inteligencia.”

Japanese proverb

Abstract

Application-Layer Traffic Optimization (ALTO) is a recently standardized protocol that provides abstract network topology and cost maps in addition to endpoint information services that can be consumed by applications in order to become network-aware and to take optimized decisions regarding traffic flows. In this work, we propose a public service based on the ALTO specification using public routing information available at the Brazilian Internet eXchange Points (IXPs). Our ALTO server prototype takes the acronym AaaS (ALTO-as-a-Service) and is based on over 2.5GB of real BGP data from the 25 Brazilian IX.br public IXPs. We evaluate our proposal in terms of functional behaviour and performance via proof-of-concept experiments, which point to the potential benefits of applications being able to take smart endpoint selection decisions when consuming the developer-friendly ALTO APIs.

Keywords: Routing (Computer network management); IXPs (Internet exchange points); Computer networks; SDN (software defined networking).

Resumo

Otimização de Tráfego na Camada de Aplicação (ALTO - *Application-Layer Traffic Optimization*) é um protocolo recentemente padronizado que fornece uma topologia da rede e mapa de custos abstratos, além de serviços de informação de endpoints que podem ser consumidos pelos aplicativos, a fim de tornar-se conscientes da rede e tomar decisões otimizadas sobre os fluxos de tráfego. Neste trabalho, propomos um serviço público baseado nas especificações ALTO usando informação de roteamento pública disponível nos Pontos de Troca de Tráfego (PTTs) brasileiros. Nosso protótipo de servidor ALTO, representado pela sigla AaaS (ALTO-as-a-Service), é baseado em mais de 2,5 GB de dados BGP reais dos 25 PTTs públicos brasileiros (IX.br). Nossa proposta é avaliada em termos de comportamento funcional e desempenho através de experimentos de prova de conceito que apontam como potencial benefício das aplicações, a capacidade de tomar decisões inteligentes na seleção de endpoint ao consumir as APIs ALTO.

Palavras-chaves: Roteamento (Administração de redes de computadores); Engenharia de tráfego; Redes de computadores; Bancos de dados.

List of Figures

List of Tables

Acronyms

AaaS ALTO-as-a-Service. 21, 47, 61–64

ALTO Application-Layer Traffic Optimization. 20, 21, 23–25, 28, 29, 32–36, 38, 39, 48, 49, 61, 64

AS Autonomous System. 20, 21, 26–28, 38, 47, 48, 60, 61

ASN Autonomous System Number. 20, 26, 38, 56, 60, 64

CC Correlation Coefficient. 59, 60

CDN Content Delivery Network. 19, 24, 29

CI Confidence Interval. 58, 59

CL Confidence Level. 58–60

ECS Endpoint Cost Service. 25, 64

EPS Endpoint Property Service. 25, 64

FCC Federal Communications Commission. 28, 31

GDB Graph Database. 35–40

GTC General Terms and Conditions. 26

ISP Internet Service Provider. 19, 20, 23, 57, 64

IX.br Internet Steering Committee project in Brazil. 27

IXP Internet eXchange Point. 20–22, 26–28, 32, 36, 38, 47, 48, 53, 56, 63, 64

LG Looking Glass. 28, 32, 36

MBA Measuring Broadband America. 28, 31

NAP Network Access Point. 25

NSF National Science Foundation. 25

NSP Network Service Provider. 23, 25

ODL OpenDaylight. 39

P2P Peer-to-peer. 19

P4P Provider Portal for Applications. 30

PID Provider-Defined Identifier. 20, 25, 33–35, 37–40, 64

RA Routing Arbiter. 25

SDN Software-Defined Networking. 39, 64

Contents

1 Introduction

Internet applications such as file sharing, real-time communications and those served from Content Delivery Networks (CDNs) rely on some sort of network topology and cost/performance information to select the best nodes in order to optimize data transfer. Each application, however, uses different means to build such overlay maps without taking into account underlying network topology considerations or network operator insights. Thus, the application endpoint selection is commonly performed based on partial and inaccurate network views or even randomly in some cases, impacting both the application performance and the efficient use of the networking infrastructure (SEEDORF; BURGER, 2009).

Aiming to fill this gap, the Application-Layer Traffic Optimization (ALTO) protocol (ALIMI *et al.*, 2014) was designed to allow network operators to provide abstract network information to Internet applications, which may use this information to optimize their connectivity decisions in alignment with the network operators' cost interests and traffic engineering practices. The network information is conveyed in the form of abstract Map Services (Network Map and Cost Map) by an ALTO server (see Fig. 1). A Network Map divides all endpoints (e.g., IPv4/IPv6 addresses or prefixes) in Provider-Defined Identifiers (PIDs) and a Cost Map provides the cost between each pair of PIDs so that it is possible to have a ranking of priority or preference among any pair of endpoints.

Most of ALTO implementations today are created by Internet Service Providers (ISPs) based on their individual knowledge of their network dynamics and on the costs associated with peering and transit links (GURBANI *et al.*, 2014). However, recent efforts (GURBANI *et al.*, 2014) show that third parties (not associated with ISPs) can also create valuable Network and Cost Maps from public information.

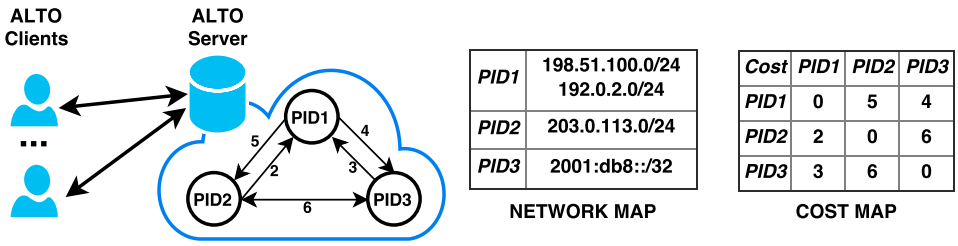


Figure 1 – Concept of ALTO and two main services: Network Map and Cost Map.

1.1 Research Problem and Objectives

In this section, we describe problems in distributed network applications in scope of our research along our objectives to tackle these challenges.

1.1.1 Problem Description

Over the years, the global Internet traffic has increased. By 2015, for example, the annual global Internet traffic[1] surpassed 620 Exabytes (52.0 Exabytes per month), a rise of more than 18 percent compared to 2014 (see Table 1). The Zettabytes (1000 Exabytes) threshold will be surpassed by the end of 2018 and the amount of traffic in 2019 will reach 136.1 Exabytes per month, a three-fold increase in comparison with 2014. If an attempt is made to divide all this Internet traffic into two segments, we could consider: (i) the Consumer Traffic (Fixed Internet and Mobile Data) generated by households, university populations, and Internet cafés representing over 80 percent of the total Internet traffic between 2016 and 2019 and (ii) the Business Traffic, now accounting for 20 percent of the global Internet traffic, including traffic generated by businesses and governments that crosses both the public Internet and mobile access points (INDEX, 2015).

Table 1 – Global Internet Traffic by Segment and by Network Type (Exabytes per Month), 2014-2019. Adapted from (INDEX, 2015).

Segment	**Network Type**	**2014**	**2015**	**2016**	**2017**	**2018**	**2019**	**CAGR (*) 2014-2019**
Consumer	Fixed Internet	31.5	37.9	46.5	58.1	72.9	91.0	24%
	Mobile Data	2.1	3.4	5.6	8.9	13.6	20.5	59%
Business	Fixed Internet	8.4	9.9	11.8	14.1	17.2	20.9	20%
	Mobile Data	0.5	0.7	1.2	1.7	2.5	3.7	51%
TOTAL Internet Traffic		**42.4**	**52.0**	**65.1**	**82.9**	**106.2**	**136.1**	

(*) Compound Annual Growth Rate (CAGR) from 2014 to 2019.

Taking into account the Consumer Traffic, Figure 2 gives an idea of its distribution by application category between 2014 and 2019 (Exabytes per Month):

- **Internet video:** Includes Internet-video-to-TV (e.g., Netflix), short-form and long-form Internet video (e.g. YouTube and Hulu respectively), live Internet video, online video purchases and rentals, webcam viewing, and web-based video monitoring (excludes P2P video file downloads).
- **Web, email, and data:** Includes web, email and other data traffic (excludes file sharing).
- **File sharing:** Examples: BitTorrent and eDonkey, as well as web-based file sharing.
- **Gaming:** Includes casual online gaming, multiplayer virtual-world gaming, and networked console gaming.

[1] All IP traffic that crosses an Internet backbone. IP traffic that remains within the corporate WAN and IP transport of TV and Video on Demand [VoD] are not considered.

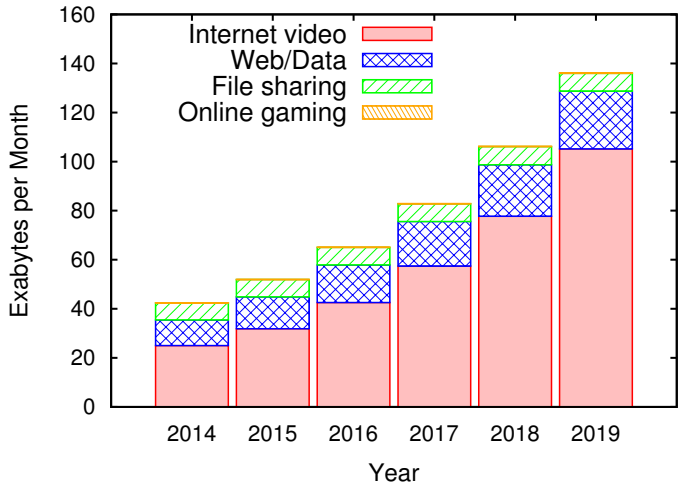


Figure 2 – Distribution of Global Internet Traffic by Type of Application. Adapted from (INDEX, 2015).

It is important to note that the sum of all forms of IP video (Internet video and video-streamed gaming) along with file sharing applications are, and will continue to be over 86 percent of the global consumer IP traffic. Clearly, this type of applications — mostly video-streaming services— enjoy a surge of popularity, and one of the principal methods to deliver these services are CDNs. In this context, P2P systems and real-time communications provided through CDNs are some Internet-distributed applications operating under a similar mechanism which enables them choose to which endpoint to connect. An endpoint could be, for example, a user with shared resources, such as files or media relays in a file-sharing application or cache/mirror servers with software repositories or rich media content in a CDN.

Mostly, the endpoint selection process in many Internet distributed applications is programmed so that available peers are randomly selected from a large set with which data is exchanged. As a result of this sub-optimal selection, for example, a large amount of inter-ISP traffic is generated, which is more costly for the network and provokes the congestion of applications.

There are currently techniques such as MPLS (ROSEN *et al.*, 2001) and Diff-Serv (GROSSMAN, 2002) to help solve the problem of network traffic optimization. However, distributed applications cannot directly use traditional traffic optimization techniques, because these mechanisms are based on requirements such as low latency, high reliability and priority that are usually implemented at the link and network layers, tending to be almost transparent.

Currently, applications do not directly use network information but attempt to infer about the network state by means of per application overlay measurements. While hiding network details (e.g. network topologies, link availability, routing policies, path costs) is a key requirement for network operators such low-level and network-centric, specific information would not be ease consumed by applications. Network applications could benefit from such

network-level information provided the information is at the right abstraction level and can be easily consumed. A "dialogue" between applications and the network could be benefitial for both parties contributing to a more efficient use of resources and increasing the application performance. However, constrained knowledge of the underlying network topology based only on localized views from the point of view of a single ISPs limits the potential and scope for application-laver traffic optimization.

1.1.2 Objectives

The main objective of this work is to create and provide Application-Layer Traffic Optimization (ALTO) services based on public information, more specifically, public routing information at Brazilian Internet eXchange Points (IXPs). ALTO implementation considerations today focus on use cases where an ISPs creates and delivers ALTO information. In this work, we intend to address the issues identified in the previous section by exploring the ALTO protocol ability to be used by user communities or third parties not associated with the ISP to provide useful and consumable topology and cost map information.

To achieve this, our service must:

- Provide a mechanism for offering peers the information that enables the selection of the nearest neighbors rather than making a random selection. This will result in a better experience for the user and it can also be less costly for the network operators.

- Provide an external service with knowledge of underline network topology, so as to allow connecting with the best hosts that contain the desired resource(s) (e.g., a chunk of a file, a file, media relay, etc.). In this manner, the application performance could have a positive impact in terms of bandwidth, delay, download time, etc.

1.1.3 Approach and Contributions

In this work, we propose a public service called ALTO-as-a-Service (AaaS) that leverages the routing information openly available at Internet eXchange Points (IXPs). Different from related works such as (MADHUKAR; WILLIAMSON, 2006), which provides ALTO services through the use of ISP's policies (not available to outside parties), AaaS creates operator-neutral and public ALTO services based on the BGP routing data from public IXPs operating in Brazil by the IX.br project (BRITO *et al.*, 2015). The raw BGP data is converted into ALTO information, and then stored in a Graph Database (GDB) to be then delivered to ALTO clients through an ALTO server via RESTFul APIs. In order to define the Network Map, each Autonomous System Number (ASN) represents a PID and every prefix (IPv4/IPv6) advertised by an Autonomous System (AS) correspond to an endpoint. Based on this Network Map, it is pos-

sible to create different Cost Maps based on the physical or the AS-level topological distance between each pair of ASes.

We evaluate AaaS regarding its functional behaviour (i.e. compliance to the ALTO protocol specifications) and performance profile of our proof-of-concept prototype implementation based on the OpenDaylight network controller. In addition, we run a series of experiments by using Mininet topologies that reflect the BGP-based AS-level connectivity and a real P2P scenario in order to validate the concept and to show the potential benefits of applications using the abstract network information from the IXP routing views easily consumable through developer-friendly ALTO APIs.

The contributions of this thesis have led to four publications (see Appendix A) and can be summarized as follows:

- Methods to generate ALTO information based on public routing data at IXPs.
- Prototype implementation of our ALTO-as-Service proposal based on best of breed open source technologies (OpenDaylight and Neo4j graph DB).
- Experimental evaluations using the prototype implementation in emulated and real-world scenarios to obtain detailed measurements (network latency and throughput, AS-level hops, total download rate, download rate per peer, etc.) and evaluate the potential benefits of our proposed approach.
- Release of all source code and datasets to ensure that our experiments can be reproduced. A Python-based tool was developed to generate AS-level topologies for the Mininet emulator taking BGP data as input.

1.2 Organization

This thesis is structured into six chapters as follows. Chapter 2 provides an overview of the ALTO protocol and includes a primer on IXPs. Chapter 3 presents the proposed architecture (AaaS) along with the basic workflow. The prototype implementation based on BGP data from IX.br is described in Chapter 4. Chapter 5 validates and evaluates the proof of concept implementation with three different types of experiments: (*i*) functional, (*ii*) performance, and (*iii*) use case scenarios. Finally, we conclude the thesis in Chapter 6 and point to our future work.

2 Literature Review

In this chapter, we review relevant literature for our research proposal. First, the ALTO architecture is exposed in depth, along with information regarding how an IXP works and how it may serve as a source of network information. Then, we present some related works about how to create AS-level topology from BGP routing data at IXPs, and finally some solutions similar to our approach are described.

2.1 Background

2.1.1 ALTO Protocol

The information presented in this section about the ALTO protocol is basically referenced by the RFC - Proposed Standard document Application-Layer Traffic Optimization Protocol (RFC 7285), the RFC - Informational document Application-Layer Traffic Optimization Problem Statement (RFC 5693) and, the Active Internet-Draft document ALTO Deployment Considerations (draft-ietf-alto-deployments-14).

2.1.1.1 Definition

Application-Layer Traffic Optimization (ALTO) (ALIMI *et al.*, 2014) is a recently Internet Engineering Task Force (IETF) standardized protocol (RFC 7285) with the main goal of exposing network information, so that the applications can optimize their endpoint selections and make informed decisions on questions such as: provided a source IP_{src}, which IP_{dst} endpoints are the best[1] among n candidate destinations.

At a high level, ALTO is an information-publishing interface that fills the gap between networks and applications, allowing network operators to publicly expose abstract network information. This network-to-application information flow benefits both the network providers (i.e. ALTO information providers), who can make a better use of their networking infrastructure –provided applications base their endpoint decisions following the ALTO Cost Maps– and the applications (ALTO information consumers), which do not need to reverse engineer the network and each of them builds its own topology maps and endpoint performance rankings.

It is important to stress that, while the ALTO protocol may provide dynamic network information, it is not intended to replace near-real-time congestion control protocols (ALIMI *et al.*, 2014). In others words, network information, that can change rapidly (e.g. transport-layer congestion), is better suited to be conveyed using in-band techniques at the transport layer

[1] The “best” according to some cost metric defined by the ALTO server provider

instead of out-of-band techniques at the application layer. ALTO-based implementations for congestion control could have out-of-date information or must be constantly retrieved by applications (ALTO clients). Rather, any network information that changes in longer time periods (e.g. topological or physical distance or maximum round-trip time between two peers) and cannot be easily obtained by applications is an ideal kind of information to be processed into ALTO information.

Finally, the ALTO protocol is based on existing HTTP implementations such as RESTful (FIELDING, 2000) interfaces for client-server interaction and JSON (BRAY, 2014) for request/reply encoding.

2.1.1.2 Architecture

As shown in the ALTO architecture illustrated in Figure 3, an ALTO server gathers network information from multiple sources, such as routing protocols, dynamic and static network information, external interfaces, and so on. This input information is then used to generate an abstract and unified view of the network (ALTO information) in the ALTO server, that, in turn, responds to ALTO client requests using the ALTO protocol. In addition, the ALTO information can be updated both dynamically (based on network conditions) or on a longer time scale (based on statically configured policies).

The ALTO design allows that each component (either an ALTO server, an ALTO client or a source of network information) can be deployed separately in multi-domain environments, i.e. the ALTO entities are located in the same network provider (e.g. an ISP domain), or in different organizations or disjoint system components (STIEMERLING *et al.*, 2016). It is even possible for an ALTO server to exchange ALTO information with other ALTO servers (within the same or in another administrative domain) in order to collect fine-grained information or to adjust exported ALTO information.

The following is a more detailed explanation of each component:

a) ALTO Server

An ALTO server is a logical entity that provides interfaces (e.g. REST-ful APIs) to consult the ALTO information services. These services may be offered by several ALTO server instances, for example, we can use more than one ALTO physical server in order to apply load balancing. In addition, the ALTO specifications allow that at least three entities can operate as an ALTO server:

- **Network operators:** An entity that has a detailed knowledge of its network topology information, such as Network Service Providers (NSPs). Usually, the source of the network information and the ALTO server are part of the same organization.

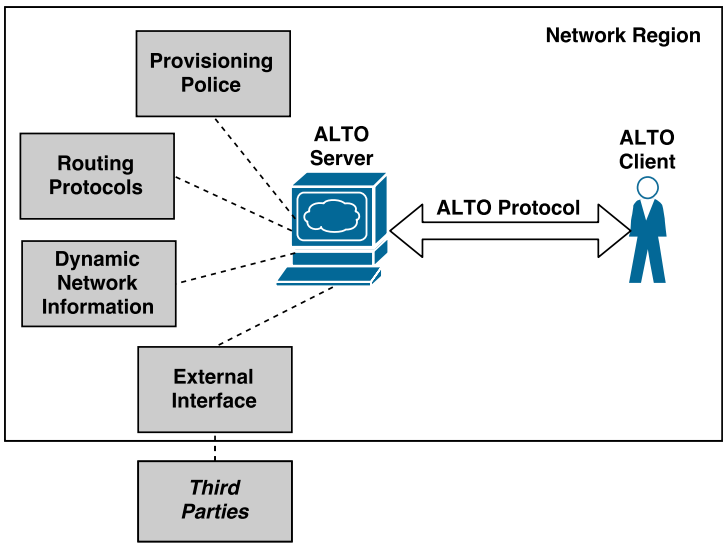


Figure 3 – ALTO Architecture. Adapted from RFC 7285.

- **Third parties:** This entity is separate from network operators; however, it could be able to retrieve network information from arrangements with network operators. For example, CDNs.
- **User communities:** Entities not associated with network providers, who may obtain network information from public data, running distributed measurement for estimating a particular topology.

b) ALTO Client

An ALTO client is a logical entity that requests ALTO queries from the ALTO server. Depending on the application architecture, an ALTO client can be situated as:

- **Resource consumer:** When the ALTO client is located on the actual host that runs the application. For example, a peer-to-peer file sharing application trying to connect to other destination peers without using a Tracker, such as edonkey.
- **Resource directory:** BitTorrent would be an example of this scenario. The Tracker acts as an ALTO client and resource consumers (peers) ask for a list of destination peers that can provide the desired resource.

c) ALTO Information Service

An ALTO server and an ALTO client exchange ALTO Information in form of ALTO Information Services (See Fig. 4). Such services include different functionalities and can be individually requested by an ALTO client in order to obtain network locations or costs for paths between network locations. The ALTO information service includes two main services (Network Map and Cost Map) and three additional ones:

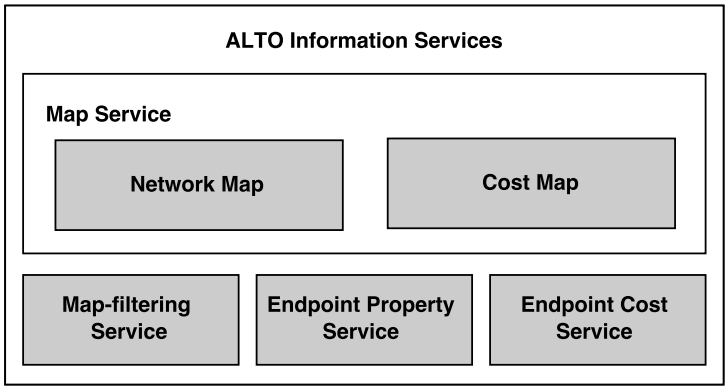


Figure 4 – ALTO Information Service. Adapted from RFC 7285.

- **Network Map:** This map represents a grouping of endpoints into PIDs that may be handled similarly based on its type, topological, physical proximity, or any other criteria. The ALTO server provider is responsible for deciding on the grouping of endpoints and the definition and semantics of PIDs.
- **Cost Map:** represents an abstract cost metric (absolute or relative) between any pair of PIDs in the form of path costs. The path cost is a custom-made cost defined and internally computed by the ALTO server implementation.
- **Map-filtering Service:** responsible for the filtering of query results on the Network Map and/or Cost Map, to narrow the reply to a subset of PIDs specified by the ALTO client.
- **Endpoint Property Service (EPS):** provides ALTO clients with information about endpoint properties (e.g. which PID belongs to a particular endpoint).
- **Endpoint Cost Service (ECS):** unlike the Cost Map (costs between PIDs), it provides information about costs between individual endpoints.

2.1.2 Internet eXchange Point

In 1992, as decided by National Science Foundations (NSFs), the transferring of the Internet core operations to the private sector was started. The result of this new commercial ecosystem was the creation of three types of operators: (i) NSPs, (ii) Network Access Points (NAPs) and, (iii) Routing Arbiter (RA). By 1997, the commercial Internet was growing exponentially, and NAPs (which provided a physical connection point between NSPs) became a problem. This is because the NAPs were populated by small operators and the largest operators wanted to peer only with large operators, which caused many of them to migrate their interconnections from NAPs to private point-to-point circuits.

By 1999, the point-to-point model between the largest operators increased linearly and the creation of interconnected circuits demanded a very high cost. In addition, these circuits were being delivered approximately one year and a half later. However, the Internet traffic

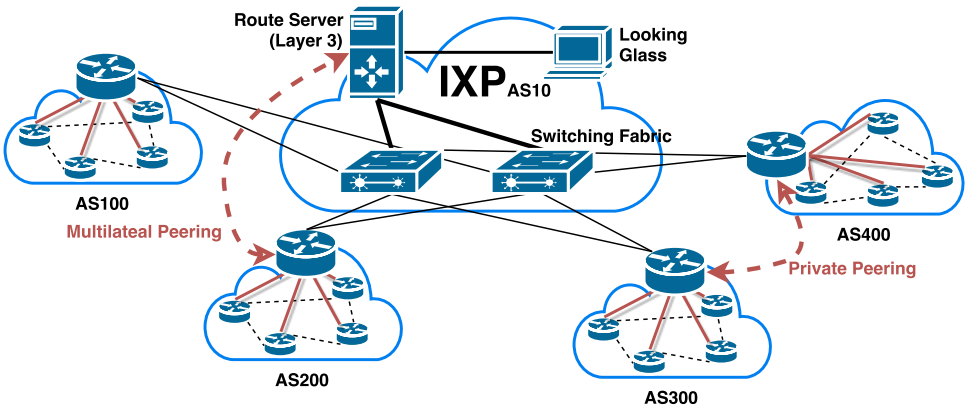


Figure 5 – Internet eXchange Point Architecture

was doubling every year, what represented a potential problem for the user's experience (NORTON, 2014). In this context, network operators considered that it made sense to create a private peering and, thus, they agreed to move into peering at carrier-neutral IXPs. This new peering ecosystem was cheaper and faster, and it is currently the dominant model for peering on the Internet.

2.1.2.1 Definition and Basic Operations

An IXP is a shared physical network infrastructure regionally installed with the purpose of facilitating the exchange of Internet traffic between more than two independent ASes and that operates below layer 3 (EUROPEAN..., 2014; CHATZIS *et al.*, 2013). With the traffic exchange as local as possible between different networks that belong to the same region, the number of hops between ASes and dependency on transit providers is reduced.

Currently, there are more than 400 national IXPs installed[2] (distributed in 139 countries around the world), and although they have different shapes and sizes, some aspects regarding their architecture and operations are applicable to most of the existing IXPs. For example, each new participant that wants to exchange traffic at an IXP is generally expected to comply with the following requirements:

- Owning and operating an Autonomous System (i.e. having an ASN).
- Establishing peering agreements with other participants.
- Implementing the BGP routing protocol.
- Agreeing to the IXP's General Terms and Conditions (GTC).

The business model —in most cases including Brazil— adopted by an IXP is open (multilateral peering), commonly allowing anyone to access a large amount of BGP public

[2] https://prefix.pch.net/applications/ixpdir/index.php. Accessed: May, 2016.

Table 2 – Traffic of some of the world's largest public IXPs (August 28, 2015). Source (BRITO *et al.*, 2015)

IXP	Country	Members	Maximum Throughput (Gbps)			Average Throughput (Gbps)		
			Daily	Monthly	Yearly	Daily	Monthly	Yearly
(01) DE-CIX	Germany	600+	3,603.10	3,854.80	3,875.10	2,375.90	2,299.20	1,964.90
(02) AMS-IX	Netherlands	731	3,620.00	-	3,872.00	2,358.00	-	2,013.00
(03) LINX	United Kingdom	630	2,472.00	2,530.00	2,575.00	1,844.00	1,631.00	1,507.00
(04) MSK-IX	Russia	384	1,409.26	1,417.01	1,569.64	924.73	788.26	778.82
(05) NL-ix	Netherlands	527	1,080.00	-	-	871.56	-	-
(06) IX.br	**Brazil**	**715**	**989.90**	**1,070.00**	**653.51**	**656.67**	**610.85**	**451.27**
(07) HKIX	Hong Kong	225	436.43	468.12	485.18	305.02	302.84	245.51
(08) SIX	USA, Canada	200	398.68	411.22	411.22	304.89	288.53	239.61
(09) JPIX	Japan	138	315.54	-	-	200.00	-	-
(10) JINX	South Africa	24	15.90	20.80	11.10	8.60	8.30	6.00

(01) http://www.de-cix.net/about/statistics/ *(02) https://ams-ix.net/technical/statistics* (03) https://www.linx.net/pubtools/trafficstats.html *(04) http://www.msk-ix.ru/network/traffic.html* *(05) https://www.nl-ix.net/network/traffic/* *(06) http://ix.br/cgi-bin/all* *(07) http://www.hkix.net/hkix/stat/aggt/hkix-aggregate.html* *(08) http://www.seattleix.net/agg.htm* *(09) http://www.jpix.ad.jp/en/technical/traffic.html* *(10) http://stats.jinx.net.za/showtotal.php*

information through telnet connections to Looking Glass (LG) servers (See Fig. 5). IP control plane information, such as the BGP routing tables, the list of BGP AS-PATH and the community codes, to give only a few examples, can be retrieved by accessing the LGs. This open access to the public routing information is the main input to build the ALTO maps proposed in our work.

2.1.2.2 Brazilian IXPs

Brazilian IXPs (BRITO *et al.*, 2015) are part of the Internet Steering Committee project in Brazil (IX.br)[3], which was created to promote infrastructure and operational means to increase the connectivity between AS networks (commercial or academic) of Brazilian metropolitan regions interested in fostering local Internet traffic exchange. With this new model, a reduction of operational cost emerges as the major advantage on Brazilian networks, since the traffic is sent from the source AS to the IXP and from this point directly to the destination AS (or as close as possible) without having to go through third-party networks, often physically distant and with international Internet connectivity.

The main features in order to implement a proper IX.br are:

- Independence from commercial operators.
- Low latency with efficient exchange of traffic.
- Low cost with high level of availability.

[3] http://ix.br

- Unique matrix for regional traffic exchange.

Currently, considering 25 IXPs in operation and 16 new locations under study[4], IX.br is the largest IXP ecosystem in Latin America and it is among the world's top ten, both in number of members and in maximum throughput rate (See Table 2).

2.2 Related Work

The work in (KHAN *et al.*, 2013) shows that it is possible to build the current view of the Internet AS-level topology from the BGP route announcement of AS by using the LG servers. The authors collected raw data from 245 LG servers across 110 countries and built the AS topology. After that, they compared that topology to other AS topology based on passive measurements using BGP routing tables (IRL[5]), and active measurements using traceroute (Ark[6] and iPlane (MADHYASTHA *et al.*, 2006)) and Internet Routing Registry (IRR[7]). Among other results, they observed 11,000 new ASes links and 686 new ASes. The main lesson learned from such experiment is that LG-based methods are less error prone than traditional traceroute-based ones, allowing raw data collection and building the AS topology in a more reliable way. In the same way, authors in (BRITO *et al.*, 2015) created AS-level connectivity graphs of all the public IXPs in Brazil (25 members) from the BGP information retrieved via telnet access to LG servers. Subsequently, the AS-level topology was analyzed and some results were highlighted, such as the average adjacency of IXP's members, the isolated and accumulated depth of paths in advertised routes, the density of peers, etc. Finally, as emphasized by the authors, this represented one of the first (if not the first) efforts to comprehend the ecosystem operation of Brazilian IXPs.

As shown in Fig. 3, the ALTO architecture allows external interfaces, so that third-parties be able to feed an ALTO server. The work in (GURBANI *et al.*, 2014) demonstrates how outside parties can create a network topology and cost maps for ALTO from public sources of information, specifically using the United States Federal Communications Commission (FCC) public database from the Measuring Broadband America (MBA) program. The authors compiled over 1 billion records spread over 90 GBytes. After examining the raw data using financial engineering and social network analysis, they provided the network topology and two cost maps (cost map for upload bandwidth and cost map for latency), each one specific for a type of application. Similar efforts, e.g. (PINTHONG; LILAKIATSAKUN, 2013; GUANXIU *et al.*, 2013) explore the ALTO protocol to create abstract topology maps based on BGP update information. They propose an approach where is possible to reduce the amount of costly inter-domain traffic generated by a BitTorrent client (in a P2P file-sharing system) when it is aware of

[4] http://ix.br/localidades/novasmap
[5] http://irl.cs.ucla.edu/topology
[6] http://www.caida.org/projects/ark
[7] http://www.irr.net

the underlying network topology. Finally, for the experimental evaluation, they use well-known P2P simulators such as: (i) PeerSim (MONTRESOR; JELASITY, 2009) and (ii) GPS (YANG; ABU-GHAZALEH, 2005).

Another work about ALTO where the information is persistent storage is described in (SHIBUYA *et al.*, 2011). The authors proposed an ISP-friendly approach based on an ALTO-like server to determine the ranking of candidate peers. The n-Tracker uses a SQL-based database for storing the Prefix List[8] and the Priority Map[9], created from static information (ISP policy information) and from dynamic information (BGP routing information). Once the Policy set function — Prefix List and Priority Map — is created, the Peerlist function can be executed. For this purpose, ALTO clients request the Peerlist (including candidate peers) to the n-Tracker, which in turn executes three sub-processes to create the Peerlist: (i) search requested peer SID, (ii) search candidate peer SID and (iii) select high priority candidate peers. Finally, the Peerlist can be sent to the ALTO client. The Prefix List and the Priority Map are updated if the policy information files changed or when the n-Tracker receives BGP UPDATE information files.

An ALTO implementation coupled with a cloud management system is presented in (SCHARF *et al.*, 2012), demonstrating how the ALTO protocol can be used to orchestrate and expose information in distributed clouds, in order to improve the user's experience in carrier clouds[10]. This proposal allows to dynamically update ALTO cost maps based on live measurements of network properties (delay and loss), using a Network Monitoring Tool. Furthermore, the proposed architecture uses two additional components: Cloud Management System and Web Portal. The Management System gets information about the internal network topology from the ALTO server, and it uses that information to manage the resources using an open source software system for creating, managing, and deploying infrastructure cloud services (CloudStack). The Web Portal provides two views: (i) a graph of the network, showing the computing nodes and communication links and (ii) a dashboard, showing a tabular view of the Virtual Machines (VMs) and computing nodes (hosts).

A solution proposal to deal with suboptimal choices in distributed applications is shown in (CHOFFNES; BUSTAMANTE, 2008). This paper retrieves network information (static and dynamic) from CDNs and provides peer selection with neither additional infrastructure nor cooperation between ISPs and subscribers. Using results collected from BitTorrent clients (over 120,000 users located in more than 100 countries), authors show that the traffic that crosses network boundaries (inter-Domain traffic) can be reduced over 33% of the time. Furthermore, destination peers selected under their approach have lower latency, lower loss rate and higher average download rate than those picked at random in a native P2P application. (BONAVENTURE *et al.*, 2008) is another work where Internet applications use a distributed request-response service to decide the best path(s) among a set of candidate paths.

[8] List where the mappings between prefixes and identifiers called SIDs (Segment ID) are registered.

[9] Matrix where the priority values between each pair of SIDs are registered.

[10] Type of solution offered by network providers to deliver cloud services running on their own network.

This informed path selection service can rank paths based on several criteria, such as routing information (e.g. BGP, OSPF/ISIS), active and passive measurements (e.g. delay, bandwidth, loss, etc.) and, custom polices defined by a network administrator. In the same vein, Provider Portal for Applications (P4P) defines a framework to allow for more effective cooperative traffic control between network providers and P2P applications (XIE *et al.*, 2008). P4P delivers network information (e.g. static network policy, network capabilities, etc.) through portals (called iTrackers) operated by network providers. This information will eventually be retrieved by peers or resource directories (e.g. a P2P Tracker) to apply a ranking preference of the endpoints, providing a desired resource.

Table 3 summarizes the related work in each of the identified objectives and includes a brief descriptions of each prior work.

Table 3 – Summary of Related Work

Contribution / Objective	**Reference**	**Description**
BGP Information from LG servers	(KHAN *et al.*, 2013)	Compares LG server-based AS topology against BGP (IRL), traceroute (Ark and iPlane), and IRR-based AS topologies.
	(BRITO *et al.*, 2015)	It was one of the first (if not the first) efforts to comprehend the ecosystem operation of Brazilian IXPs. Some results: AS-level connectivity graphs, Peering density, Depth of AS-PATH, etc.
Create and/or Provide ALTO Information	(GURBANI *et al.*, 2014)	Creates network topology and cost maps for ALTO, using public sources of information, specifically using the United States FCC database from the MBA program.
	(PINTHONG; LILAKI-ATSAKUN, 2013), (GUANXIU *et al.*, 2013)	Explore the ALTO protocol by mapping IP addresses to an AS, to create a network map and BGP route announcements from ASes to create a cost map. Both maps were created by using simulation platforms (PeerSim and GPS, respectively).
	(SHIBUYA *et al.*, 2011)	Describes n-Tracker as a server that determines the ranking of the candidate peers (in a P2P architecture) sent from an ALTO client (End users or Tracker). It uses a SQL-based database for storing the Prefix List and the Priority Map, created from static and dynamic information.
	(SCHARF *et al.*, 2012)	Presents an integrated solution between a cloud management system and ALTO (Cost Maps based on live measurements). Authors show that ALTO is well-suited to provide infrastructure-to-application information in order to improve the user's experience in a distributed carrier cloud.
Provide Network Topology Information	(CHOFFNES; BUSTAMANTE, 2008), (BONAVENTURE *et al.*, 2008), (XIE *et al.*, 2008)	P2P mechanism to find reliable information source to apply better-than-random peer selection, but not in the form of ALTO services (i.e. difficult to deploy globally without an open standard).

3 Design of ALTO-as-a-Service

We propose to deliver ALTO as a public service, which can be useful for any application interested in obtaining information about AS-level network maps for IP endpoints. The main goal is to generate ALTO PIDs (Network Map) along with a ranking of candidate PIDs (Cost Maps) using the publicly available BGP routing information at IXPs. To this end, we collect BGP tables (IPv4 and IPv6) and parse the BGP AS-PATH attributes to create the ALTO Network Map and the Cost Map respectively. The resulting data structures are stored in a graph-based database (Neo4j) and finally delivered as ALTO services through HTTP Rest APIs.

Figure 6 illustrates the AaaS workflow: (a) Acquiring Input Data, (b) Building Graph Data Models, (c) Creating ALTO Information, and (d) Delivering ALTO Services to serve ALTO client requests. Examples of ALTO clients include, but are not limited to, (1) host running a peer-to-peer file sharing application, (2) tracker in peer-to-peer file-sharing applications, or (3) Software Defined Networking (SDN) controllers.

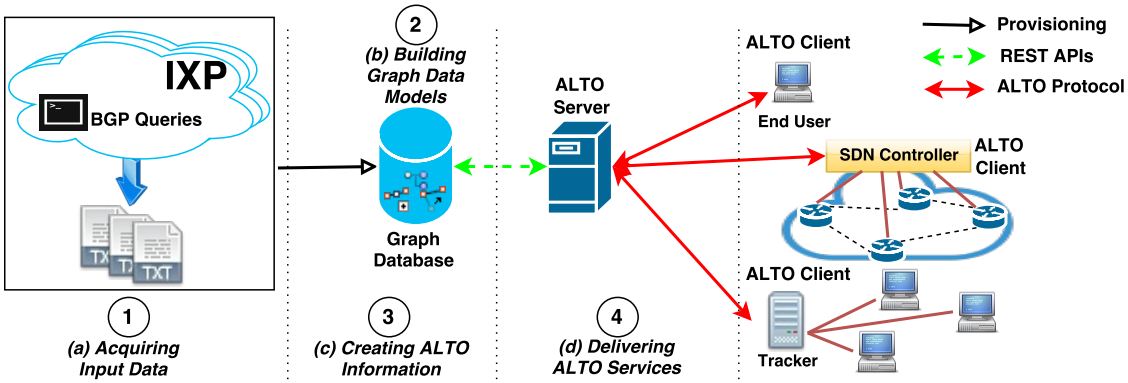


Figure 6 – AaaS PoC Workflow

3.1 Acquiring Input Data

This first step is to collect BGP routing data publicly available through LG servers, using the same methodology proposed in our IX.br ecosystem anatomy (BRITO *et al.*, 2015). Figure 7 shows the internal workflow of this process:

1. Each IXP is publicly accessed via telnet access to LG servers.
2. Once connected at each individual IXP, BGP commands are executed in order to get:
   - The control plane IPv4 and the IPv6 BGP table (*show ip bgp* and *show ipv6 bgp* commands).

- The summary list of BGP AS-PATH (*show ip bgp paths* command).

3. The third step is to locally store the raw dataset generated as output of these BGP queries in text file format.

4. The BGP raw data is then pre-processed (formatting, filtering, and assembling) to facilitate its transformation into ALTO data structures.

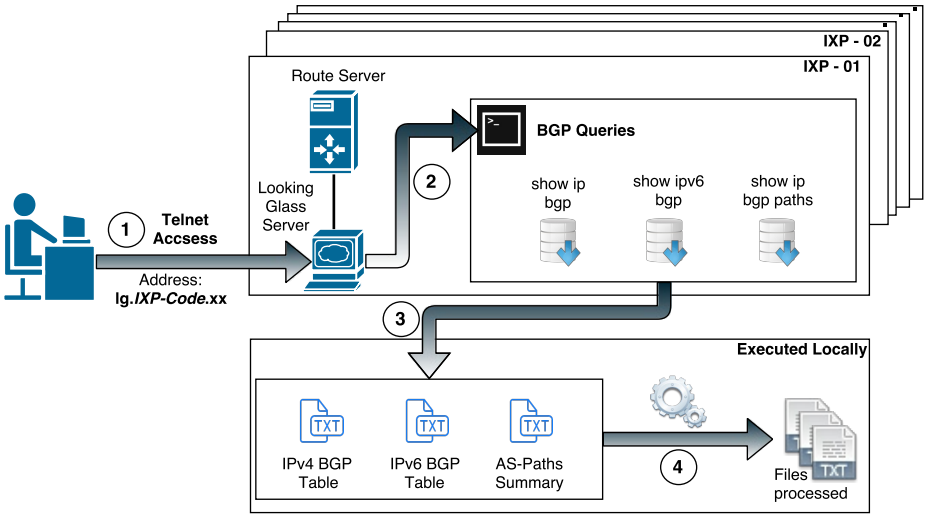


Figure 7 – Workflow of Gathering Input Data

3.2 Building Graph Data Models

This process consists of building a suitable connected graph of nodes and relationships to model the ALTO information according to the protocol specification [RFC 7285]. We opt for a property graph[1] since it provides a natural modeling approach to inherent native graph problem at hand. In addition, this approach eases the implementation of this model using a Graph DB (Neo4j), as we detail in the following section. Figure 8 provides an overview of the components used in the graph, where nodes are entities that are connected by describing their interactions, i.e. the relationships.

Detailed information about the graph nodes and relationships, along with their respective properties are presented bellow:

3.2.1 Nodes

- **Provider-Defined Identifier (PID):** An ALTO network map defines a grouping of network endpoints, and PIDs are a customized way to specify an aggregation of network endpoints. This node is labeled in model as “PID”.

[1] A graph where (i) vertices and edges can have any number of key/value properties, (ii) there can be many types of relationships between vertices and (iii) edges have a directionality.

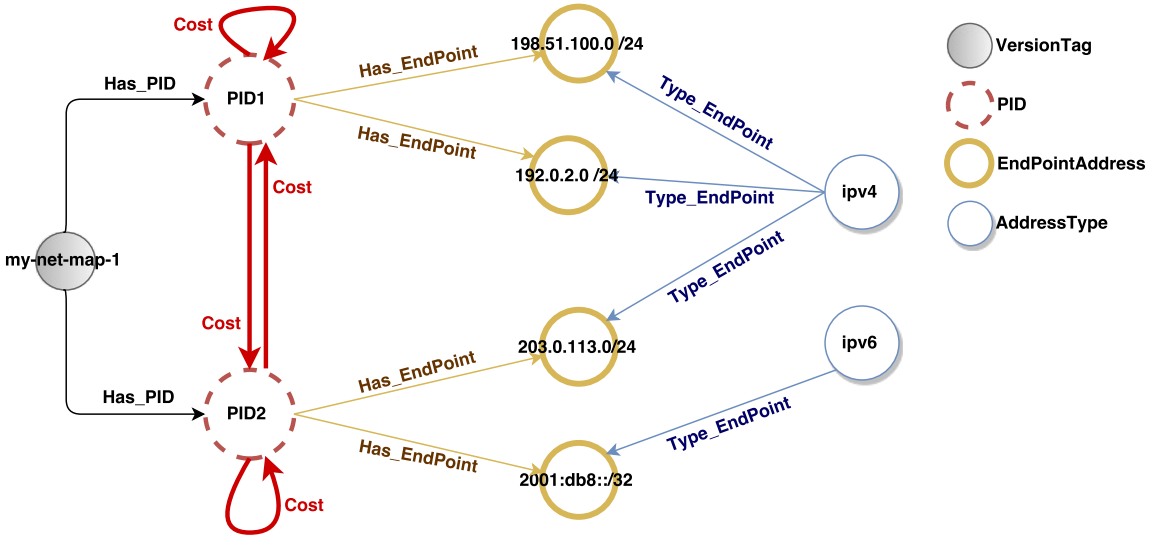


Figure 8 – Neo4j Graph Data Model based on RFC 7285 (ALIMI *et al.*, 2014)

Properties:		
RFC Name	**GDB**	**Description**
PID Name	Name	Name of a PID

- **Endpoint Address:** It used to denote a set of IP addresses and/or endpoint prefixes that belong to a particular PID. This node is labeled in our model as “EndPointAddress”.

Properties:		
RFC Name	**GDB**	**Description**
Prefix	Prefix	IP address and some indication of the mask length

- **Address Type:** When an Endpoint Address is used, we need to explicitly indicate its type. This node is labeled in our model as “AddressType”.

Properties:		
RFC Name	**GDB**	**Description**
Type	Type	“ipv4” to refer to IPv4 Endpoint Addresses or “ipv6” to refer to IPv6 Endpoint Addresses

- **Version Tag:** It is used to indicate the version of a particular resource. For example, in a network map if there is an ALTO server adding, deleting or redefining PIDs, the Version Tag should be updated. This node is labeled in our model as “VersionTag”.

Properties:

RFC Name	GDB	Description
Resource ID	ResourceID	ID unique for a resource
Tag	Tag	Version for a resource

3.2.2 Relationships

Relationships always have a start node, an end node, a direction, a type (label), and may optionally have properties. Table 4 shows the relationships used in our model.

Table 4 – Relationships and pairs of nodes that connect them.

Name	Start Node	End Node	Description
Has_PID	VersionTag	PID	To know the Version to which a particular PID belongs
Has_EndPoint	PID	EndPointAddress	For grouping Endpoint Addresses to a particular PID
Type_EndPoint	AddressType	EndPointAddress	To indicate the type (IPv4 or IPv6) of Endpoint Address
Cost	PID	PID	To determine the path cost between two PIDs

Noteworthy is the relationship labeled as "Cost", where properties can be included to create different path costs between two PIDs resulting in different Cost Map rankings. To mention just a few examples: topological distance, physical distance or round-trip time (RTT).

3.3 Creating ALTO Information

Once the input dataset and the data model are ready, the next step is to create the ALTO information and to populate the graph DB. In this step, the ALTO server administrator uses the BGP routing information retrieved to (*i*) create grouping of prefixes into PID (Network Map) by ASes, IXPs, BGP communities, points of presence, just to cite a few examples; and (*ii*) define the preferences / costs between the groups PID (Cost Map) expressed on a path cost, such as physical distance between IXPs, topological distances between ASes, among others.

3.4 Delivering ALTO Services

The last step is to deploy an ALTO web server. This is attained by implementing a client-server protocol in order to deliver the REST/JSON APIs to ALTO clients, as defined by RFC7285. Internal interfaces to retrieve ALTO information from the GDB are also necessary and we opt to converge on REST/JSON for convenience.

4 Prototype Implementation

We now turn our attention to the implementation choices and prototype details of the AaaS proof of concept. In this section, we provide further details on the BGP data set from the Brazilian IXPs, the Neo4j[1] graph-based database used as the back-end for the ALTO information, and the OpenDaylight[2] (ODL) controller used as ALTO server.

4.1 Input: IX.br BGP Data Set

The data collection methods based on LG remote access to each public IXP in Brazil are those described in (BRITO *et al.*, 2015). More specifically, the dataset (approx. 2.5GB) retrieved during December 2014 was used for our prototype implementation. Additional information on each Brazilian IXP as addresses of the LGs (column *"Looking Glass"*), operation region (city and state), code, average traffic exchange (column *"Gbps"*) and quantity of members can be found in Table 5. For data sharing and reproducibility purposes, the raw dataset (including all our 2015 snapshots) and all supporting codes are publicly available in our research group repositories[3].

For the pre-processing job of the raw data, we use a number of Java and R[4] based algorithms. For example, with R we convert the files of IPv4 and IPv6 BGP table into a readable format and exclude prefixes that are advertised by more than 2 ASes.[5] Using Java-based algorithms, we discard the AS-Paths (from the files of the summary list of BGP AS-PATH) which contain the ASN 20121 and ASN 26121 because they are reserved for the IX.br's LG and RS respectively and, therefore, do not participate in Internet routing. Table 6 shows the number of ASes and the number of IPv4 and IPv6 prefixes before and after the pre-processing. As we can see, the amount of discarded data is not significant.

4.2 ALTO Server Backend: Neo4j

As anticipated during the design discussion on the graph modeling approach, using a GDB was a natural choice to create the model and to embody the ALTO information as a property graph. Furthermore, the ALTO protocol uses a key-value store abstraction for JSON object coding that is very amenable for the Neo4j implementation choice. Neo4j is an open-

[1] http://neo4j.com/
[2] https://www.opendaylight.org/
[3] https://github.com/intrig-unicamp/ixp-ptt-br/ and https://github.com/intrig-unicamp/ALTO-as-a-Service/
[4] http://www.r-project.org
[5] ALTO protocol uses the longest-prefix matching algorithm to compute the mapping from Endpoints to PIDs, therefore a Network Map must not define two or more PIDs that contain an identical Endpoint.

Table 5 – Public IXPs Operating in Brazil (July, 2015)

#	City	State	Code	Looking Glass	Gbps	Members	
01	Belem	PA	BEL	lg.bel.ptt.br	0.44	14	
02	Belo Horizonte	MG	MG	lg.mg.ptt.br	2.07	33	
03	Brasilia	DF	DF	lg.df.ptt.br	2.98	30	
04	Campina Grande	PB	CPV	lg.cpv.ptt.br	0.69	10	
05	Campinas	SP	CAS	lg.cas.ptt.br	3.57	35	(*)
06	Cuiaba	MT	CGB	lg.cgb.ptt.br	0.00	9	(*)
07	Caxias do Sul	RS	CXJ	lg.cxj.ptt.br	0.08	5	(*)
08	Curitiba	PR	PR	lg.pr.ptt.br	16.10	68	(1)
19	Florianopolis	SC	SC	lg.sc.ptt.br	1.28	34	(*)
10	Fortaleza	CE	CE	lg.ce.ptt.br	2.72	29	
11	Goiania	GO	GYN	lg.gyn.ptt.br	1.06	24	
12	Lajeado	RS	LAJ	lg.laj.ptt.br	0.01	8	(*)
13	Londrina	PR	LDA	lg.lda.ptt.br	1.62	32	
14	Manaus	AM	MAO	lg.mao.ptt.br	0.02	8	(*)
15	Maringa	PR	MGF	lg.mgf.ptt.br	0.28	21	(*)
16	Natal	RN	NAT	lg.nat.ptt.br	0.26	13	(*)
17	Porto Alegre	RS	RS	lg.rs.ptt.br	20.85	117	
18	Recife	PE	PE	lg.pe.ptt.br	0.69	16	
19	Rio de Janeiro	RJ	RJ	lg.rj.ptt.br	39.22	68	
20	Salvador	BA	BA	lg.ba.ptt.br	1.47	47	(*)
21	Sao Carlos	SP	SCA	lg.sca.ptt.br	0.00	3	(*)
22	Sao Jose dos Campos	SP	SJC	lg.sjc.ptt.br	0.47	13	
23	Sao Jose do Rio Preto	SP	SJP	lg.sjp.ptt.br	0.62	11	(*)
24	Sao Paulo	SP	SP	lg.sp.ptt.br	429.45	667	(1)
25	Vitoria	ES	VIX	lg.vix.ptt.br	0.80	22	

(1) There are filters in LG compromising the BGP table.

(*) Data provided by NIC.br, since publicly access was denied.

source non-relational GDB implemented in Java, that shows high scalability and flexibility. It supports true ACID transactions, high availability, and scales to billions of nodes and relationships (NEO4J, 2015). In addition, application development is highly facilitated through the use of high-speed traversal query languages such as Cypher.

As shown in Figure 9, to populate the Neo4j GDB, Java-based programs were developed to (*i*) read the input dataset, (*ii*) create the Network and Cost Map, and (*iii*) store the final property graphs into Neo4j using REST interfaces. Next we detail (a) how the endpoints are grouped into PIDs to create the Network Map, and (b) how the path cost between PIDs is computed to create the Cost Map:

Table 6 – Number of ASes and Prefixes (IPv4/IPv6) before and after the dataset pre-processing task at Brazilian IXPs.

	Raw Data	**After Proc.**	**% Out**	**99% CL ±1 MOE**
ASes	49,586	**48,962**	1.26%	12,460
IPv4 Prefixes	563,164	**556,628**	1.16%	16,163
IPv6 Prefixes	21,666	**21,427**	1.10%	9,412

4.2.1 Network Map

For each pre-processed IPv4 and IPv6 BGP table, the developed algorithm reads each route announcement entry and extracts the ASN that originated/advertised a prefix. The ASN serves as the PID (location) grouping, resulting in a total of 48,962 PIDs (consistent with the current global amount of Internet AS). Next, each prefix (be it IPv4 or IPv6) is associated with a particular PID (i.e., AS) considering the origin of the prefix announcement.

4.2.2 Cost Map

The path cost between PIDs is calculated as the AS-level topological distance corresponding to the amount of traversing ASes, i.e. path cost equals the number of AS hops between a source and destination AS. A lower cost between PIDs indicates a higher preference for traffic. Two Cost Maps variations are proposed: one which represents the absolute topological distance, and a second one which represents the relative distance. In the latter case, hops between ASes present in the same IXPs are zeroed to favor intra-IXP traffic.

The AS-Path summary files are used to create these maps (see Fig. 9, Cost Map). First, we build the AS-level connectivity graph using an auxiliary GDB. Then, we compute the path cost between each pair of ASes using Cypher queries. Whenever more than one path between two ASes is found, the path with the least number of traversing ASes is chosen. Finally, the path cost is updated in the main ALTO GDB instance.

The Cost Map is a square matrix of order N, where N corresponds to the number of PIDs resulting in over 2.3 billion (10^9) relationships labeled as `cost` (two properties are used to distinguish between the absolute and relative distances). Hence, the process of Cost Map creation is partially completed in a proactive manner, while the remainder of the map is created on the fly (or reactively) based on ALTO client requests asking for path cost between specific PIDs.

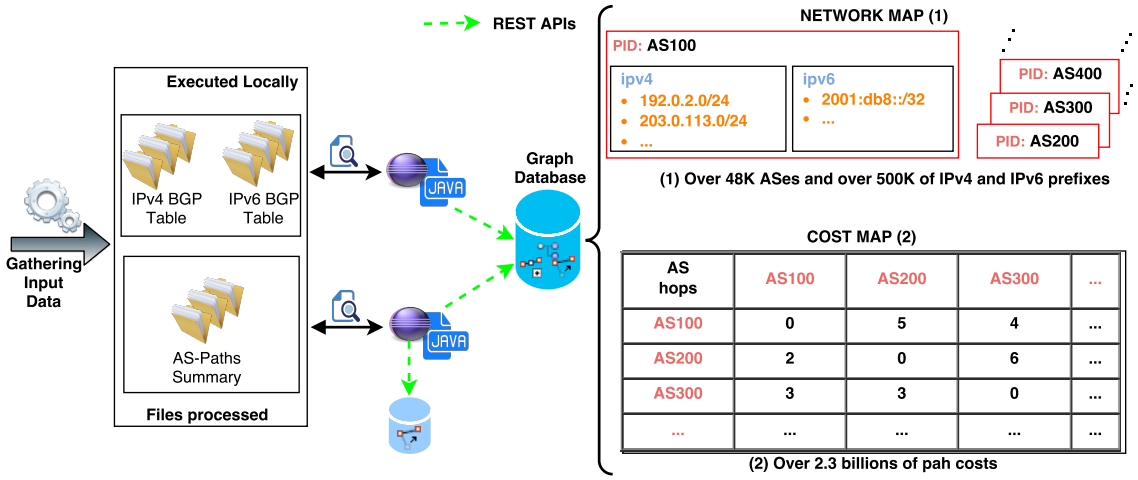


Figure 9 – The Network Map and Cost Map Map services created from the IP BGP table and AS-Path summary files, respectively.

4.3 ALTO Server Front-End: OpenDaylight

In order to deliver ALTO services, we opt to reuse the OpenDaylight (ODL) controller that features an ALTO project[6] since the Lithium software release. ODL is an open source SDN controller architecture with production quality code and proven scalability and reliability. The initial release of ALTO in ODL includes, among other modules, ALTO Northbound providing basic ALTO services as RESTful web services (Northbound APIs) for ALTO client/server communications. ALTO Northbound APIs generate ALTO services from data stored in the MD-SAL data store (an ODL core component). For our AaaS implementation, it was necessary to modify the Northbound APIs to generate ALTO services from the data stored in the Neo4j GDB (instead of the MD-SAL topology).

We checkout the *stable/lithium* branch in the ALTO project GitHub repository,[7] which implements the following ALTO services: (*i*) full Map Service, (*ii*) Map-Filtering Service, (*iii*) Endpoint Property Service, and (*iv*) Endpoint Cost Service. To accomplish our proof of concept evaluations, our initial ALTO server delivers the Map-Filtering Service, i.e. the filtered Network Map and the filtered Cost Map, which allow ALTO clients to specify filtering criteria to return only a subset of the full Map Service. Hence, those two Northbound APIs were modified so that the ALTO server retrieves information from the Neo4j backend and converts it into the ALTO format specification. Essentially, two new public functions have been added (`getNetworkMap` and `getCostMap`) in the `AltoNorthbound.java` file, replacing the original function call within each Northbound API (`retrieveFilteredNetworkMap` and `retrieveFilteredCostMap`).

The first function creates and returns a filtered Network Map (See Code 1). The code contains as input the Resource ID of the service (`id` parameter) and a list of PIDs (`filter`

[6] https://wiki.opendaylight.org/view/ALTO:Main
[7] https://github.com/opendaylight/alto

```
1  public RFC7285NetworkMap getNetworkMap(String id, RFC7285NetworkMap.Filter filter) {
2      //extract the PIDs
3      String lstPID = "";
4      for (String sAux : filter.pids)
5          lstPID = lstPID + "'" + sAux + "',";
6      lstPID = lstPID.substring(0, lstPID.length() - 1);
7
8      //Cypher query to obtain the Network Map
9      String strQuery = "";
10     strQuery =
11         String.format(new StringBuilder()
12         .append("MATCH (v:VersionTag {ResourceID:'%s'})-[r:Has_PID]->(pid)")
13         .append(" WHERE pid.Name IN [%s]")
14         .append(" OPTIONAL MATCH (pid)-[e:Has_EndPoint]->")
15         .append("(p)<-[r2:Type_EndPoint]-(type)")
16         .append(" WITH v.Tag AS Tag,pid.Name AS Name,")
17         .append(" COLLECT([type.Type,p.Prefix]) AS p")
18         .append(" RETURN Tag, Name,")
19         .append(" [prefix IN FILTER(pAux in p where pAux[0]='%s') | prefix[1]] AS IPv4,")
20         .append(" [prefix in FILTER(pAux IN p where pAux[0]='%s') | prefix[1]] AS IPv6")
21         .append(" ORDER BY Name").toString(), id, lstPID, "ipv4", "ipv6");
22
23     //Execute the Cypher query
24     REST_Query(strQuery, SERVER_ROOT_URI);
25
26     //Convert the query result (Json format) to Cost Map format (RFC7285)
27     Iterator<JsonNode> node = jsonResult.path("results").findPath("data").elements();
28     RFC7285NetworkMap nm = convert(node);
29
30     //Return the filtered Network Map
31     return nm;
32 }
```

Listing 1 – getNetworkMap function added in the `AltoNorthbound.java` file.

parameter), allowing obtain a subset of the full Network Map. This information is then used to build the Neo4j Cypher query statement and and execute it in our GDB (`REST_Query` function). Finally, the response of the POST request —in Json format— is converted to Network Map format (RFC7285) using the function.

In the case of filtered Cost Map function (See Code 2), the `filter` parameter contains: (i) two list of PIDs (source and destination PIDs) and (ii) the Cost Type (Cost Metric and Cost Mode). This parameters, together with the Resource ID, is used to build the Cypher query. After, when we execute this statement, the Neo4j GDB returns the filtered Cost Map in Json format. As a last step, results are converted from Json to ALTO type following the RFC7285 specifications.

```
1  public RFC7285CostMap getCostMap(String id, RFC7285CostMap.Filter filter) {
2      //extract the Cost Type (Cost Metric and Cost Mode)
3      String strMode = filter.costType.mode;
4      String strMetric = filter.costType.metric;
5
6      //extract the source PIDs
7      String strSRC = "";
8      for (String sAux : filter.pids.src)
9          strSRC = strSRC + "'" + sAux + "',";
10     strSRC = strSRC.substring(0, strSRC.length() - 1);
11
12     //extract the destination PIDs
13     String strDST = "";
14     for (String sAux : filter.pids.dst)
15         strDST = strDST + "'" + sAux + "',";
16     strDST = strDST.substring(0, strDST.length() - 1);
17
18     //Cypher query to obtain the Cost Map
19     String strQuery = "";
20     strQuery =
21         String.format(new StringBuilder()
22         .append("MATCH (v:VersionTag {ResourceID:'%s'})-[r:Has_PID]->(s)")
23         .append(" WHERE s.Name IN [%s]")
24         .append(" OPTIONAL MATCH (s)-[c:Cost]->(d)")
25         .append(" WHERE d.Name IN [%s]")
26         .append(" WITH v.ResourceID AS ResourceID, v.Tag AS Tag,")
27         .append(" s.Name AS PIDsrc, d.Name AS PIDdst, c.%s AS hops")
28         .append(" ORDER BY PIDsrc, hops, PIDdst")
29         .append(" return ResourceID, Tag, PIDsrc, collect([PIDdst, hops])")
30         .toString(), id, strSRC, strDST, strMetric);
31
32     //Execute the Cypher query
33     REST_Query(strQuery, SERVER_ROOT_URI);
34
35     //Convert the query result (Json format) to Cost Map format (RFC7285)
36     Iterator<JsonNode> node = jsonResult.path("results").findPath("data").elements();
37     RFC7285CostMap cm = convert(node, strMode, strMetric, strDST);
38
39     //Return the filtered Cost Map
40     return cm;
41 }
```

Listing 2 – getCostMap function added in the `AltoNorthbound.java` file.

5 Experimental Evaluation

We evaluate our proposal by carrying out three different types of experiments using the proof of concept AaaS implementation:[1] (*i*) functional behaviour (i.e. in accordance with the ALTO specifications), (*ii*) system performance profiling, (*iii*) emulated use case scenarios (IXP-based network models) and a real P2P system environment which include information about the configuration, workload, metrics and a detailed analysis of the effectiveness of our approach for the endpoint selection process.

5.1 Functional Evaluation

We evaluated whether the ALTO server delivers ALTO services in compliance with RFC 7285. For that purpose, we used a REST client tool[2] to retrieve ALTO information in JSON format, communicating with the ALTO server via HTTP request. Test cases are corresponding to the RESTful web services available in our ALTO server (the filtered Network Map and the filtered Cost Map).

5.1.1 Test Cases

The data structures and message templates in our test cases are based on the Internet-Draft document Interoperability testing of the ALTO Protocol (GUO, 2015) that provides a framework to test the functionality and interoperability of an ALTO client and an ALTO server.

a) Test-FILTER-1: An ALTO client sends a request to get a filtered Network Map of PIDs: AS53143 and AS12322.

Client -> Server

```
POST /controller/nb/v2/alto/filtered/networkmap/my-default-network-map HTTP/1.1
Host: localhost:8181
Content-Length: ***
Content-Type: application/alto-networkmapfilter+json
Accept: application/alto-networkmap+json, application/alto-error+json

{
    "pids" : [ "AS53143","AS12322"]
}
```

[1] Single server configuration Intel® Core™ I7-4790 @ 3.60GHz x 8 with 16GB RAM, running Ubuntu 14.04LTS (Linux) 64-bit.

[2] https://www.getpostman.com/

Server -> Client

```
HTTP/1.1 200 OK
Content-Length: 172
Content-Type: application/alto-networkmap+json

{
    "meta": {
        "vtag": {
            "resource-id": "my-default-network-map",
            "tag": "Oau8s4KELGkl9fgeIudhOWSSRyaebzrD"
        }
    },
    "network-map": {
        "AS12322": {
            "ipv4": [ "213.228.0.0/18", "91.160.0.0/12", "62.147.0.0/16" ],
            "ipv6": [ "2a01:e00::/26" ]
        },
        "AS53143": {
            "ipv4": [ "186.194.238.0/23" ]
        }
    }
}
```

b) Test-FILTER-2: An ALTO client sends a request to get a filtered Cost (absolute distance) Map from a source PID (AS53187) to a set of destinations PIDs (AS53143 and AS12322).

Client -> Server

```
POST /controller/nb/v2/alto/filtered/costmap/my-default-network-map HTTP/1.1
Host: 192.168.122.1:8181
Content-Length: ***
Content-Type: application/alto-costmapfilter+json
Accept: application/alto-costmap+json, application/alto-error+json

{
    "cost-type" :{"cost-mode": "Numerical",
                  "cost-metric": "HopsNumber"
                 },
    "pids" : {
        "srcs" : [ "AS53187" ],
        "dsts" : [ "AS53143","AS12322" ]
            }
}
```

Server -> Client

```
HTTP/1.1 200 OK
Content-Length: 172
Content-Type: application/alto-costmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "my-default-network-map",
        "tag": "Oau8s4KELGkl9fgeIudhOWSSRyaebzrD"
      }
    ],
    "cost-type": {
      "cost-mode": "Numerical",
      "cost-metric": "HopsNumber"
    }
  },
  "cost-map": {
    "AS53187": { "AS53143": 2, "AS12322": 3 }
  }
}
```

c) Test-FILTER-3: An ALTO client sends a request to get a filtered Cost (relative distance) Map from a source PID (AS53187) to a set of destinations PIDs (AS53143 and AS12322).

Client -> Server

```
POST /controller/nb/v2/alto/filtered/costmap/my-default-network-map HTTP/1.1
Host: 192.168.122.1:8181
Content-Length: ***
Content-Type: application/alto-costmapfilter+json
Accept: application/alto-costmap+json, application/alto-error+json

{
    "cost-type" :{"cost-mode": "Numerical",
                  "cost-metric": "HopsNumberPTT"
                },
    "pids" : {
        "srcs" : [ "AS53187" ],
        "dsts" : [ "AS53143","AS12322" ]
            }
}
```

Server -> Client

```
HTTP/1.1 200 OK
Content-Length: 172
Content-Type: application/alto-costmap+json

{
  "meta": {
    "dependent-vtags": [
      {
        "resource-id": "my-default-network-map",
        "tag": "0au8s4KELGkl9fgeIudh0WSSRyaebzrD"
      }
    ],
    "cost-type": {
      "cost-mode": "Numerical",
      "cost-metric": "HopsNumberPTT"
    }
  },
  "cost-map": {
    "AS53187": { "AS53143": 0, "AS12322": 1 }
  }
}
```

As expected, our AaaS prototype delivers the ALTO services in accordance with the ALTO specifications and fully reflects the ALTO information stored in Neo4j.

5.2 System Performance Profiling

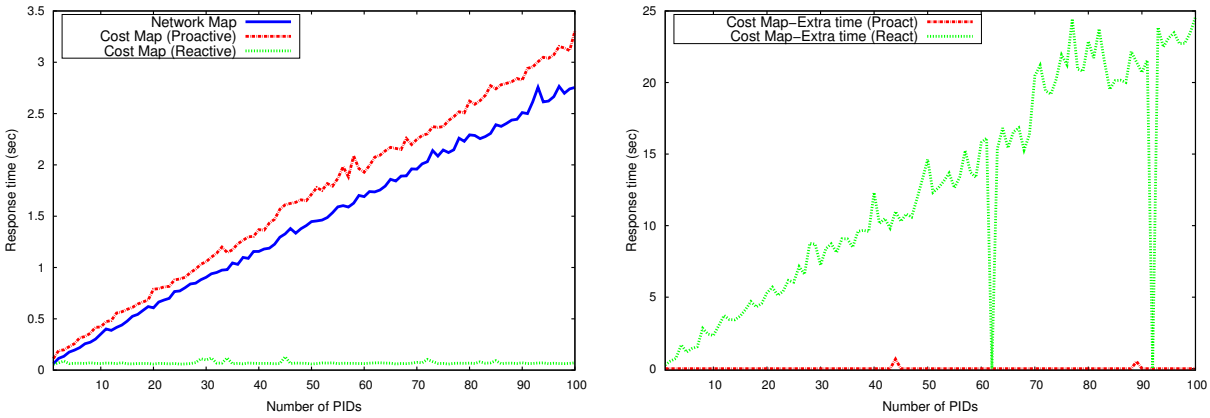
In order to assess the performance of our AaaS prototype, we calculate the response time for the filtered Network and Cost Map (absolute distance) services. For both services, between 1 to 100 PIDs[3] are randomly selected, totaling 100 requests, where each request is executed 10 times. The average transaction time is shown in Figure 10.

For the Network Map service, we can observe that the response time increases in proportion to the number of PIDs (See Fig. 10a, Network Map). For example, when the number of PIDs is 5 and 50, the average time was 0.19 Sec and 1.45 Sec, respectively.

Regarding the Cost Map service, we have two PID input parameters (see Subsection 5.1.1). A single PID is used as source PID (`srcs` parameter) and the amount of destination PIDs (`dsts` parameter) varies from 1 to 100. Another relevant factor in the response time is whether a proactive or reactive mode is evaluated.

In order to evaluate the proactive approach, we select a source PID with all possible path costs already created (about 49K path costs). As shown in Figure 10a, on average, the

[3] With 50 being the default number of candidate peers in the BitTorrent P2P application.



(a) (b)

Figure 10 – (a) Response processing time for Network and Cost Map (Absolute Distance) services and (b) processing time used to compute two additional steps: (i) calculation of number of hops and (ii) insertion of these numbers into the database.

processing time is 19.73% (or 0.24 Sec) slower compared to the Network Map service. We have a higher query cost, as there are two access operations to the database: one to retrieve the source PID and one to retrieve the path cost with destination PIDs. It is also necessary to point out that this mode does not have to spend time, i.e. 0 Sec (see Fig. 10b, Proact), to compute two additional steps (to calculate the number of hops and to insert these numbers into the database), as they were previously processed.

In the reactive mode, although it also has two access operations, the average processing time is just 0.07 Sec (see Fig. 10a, React). This behavior can be explained by the fact that the path costs are only created for destinations PIDs of the HTTP request, namely, up to a maximum of 100 path costs. However, for the first request, the time it takes to execute the two additional steps must be considered; for instance, when we send 10 destination PIDs, the processing time is around 2.3 Sec (see Fig. 10b, React). A particular case is when 62 and 92 destination PIDs are sent, we can see that it takes 0 Sec. This is because the selected destination PIDs already had the path cost created with the source PID, so no additional steps were needed. Eventually, when all possible path costs for a particular PID are already created, the response time should be proportional to the number of destination PIDs, as shown in a proactive manner.

Finally, the latency for obtaining a response from AaaS can be eased either by reusing cached Maps (at the risk of being out-of-date) or by running servers in multiple domains through the delegation of some ALTO information (e.g. Filtered Network and Cost Maps) to other subdomains.

5.3 Use Case Scenarios

We now try to assess the potential effectiveness of AaaS in delivering useful network information, so that an originating peer can make better decisions (in terms of network performance) regarding the candidate destination peers. Three validation environments are being considered in our work: (i) a first basic scenario where we begin to explore the locality awareness benefits in a P2P application, (ii) a previous extended version where we use AaaS in order to reduce cross-ISP traffic, (iii) the use of real trackers and real peers from the Internet for evaluating the performance of a P2P application (with and without AaaS). Each scenario and corresponding procedure are described next.

5.3.1 IP endpoint selection with AaaS

This first experiment is arguably simplified and mainly serve as a strawman to illustrate the ALTO potential to deliver useful services to IXP members (and/or third-party applications) to perform better-than-random peer selection. This makes it possible to obtain the nearest peer in order to improve the network application performance (for example, to minimize latency and to upgrade upload capacities or download speed).

5.3.1.1 Experimental Setup

The network model used is based on a small IXP ecosystem (see Fig. 11) consisting of 22 ASes, each represented by a switch abstraction in the Mininet emulator[4]. A sample AS-Path summary file based on real BGP data was used to create the AS-level connectivity in our experiment topology. The large AS switch represents the IXP and then 10 communicating peers are represented as Mininet hosts attached to the (AS abstraction) switches. Links between ASes follow the sample AS-Path attributes and were set with larger bandwidth and lower delay when closer to the IXP.

[4] Source code available at: https://github.com/intrig-unicamp/IXP-Brazil-Mininet-Code

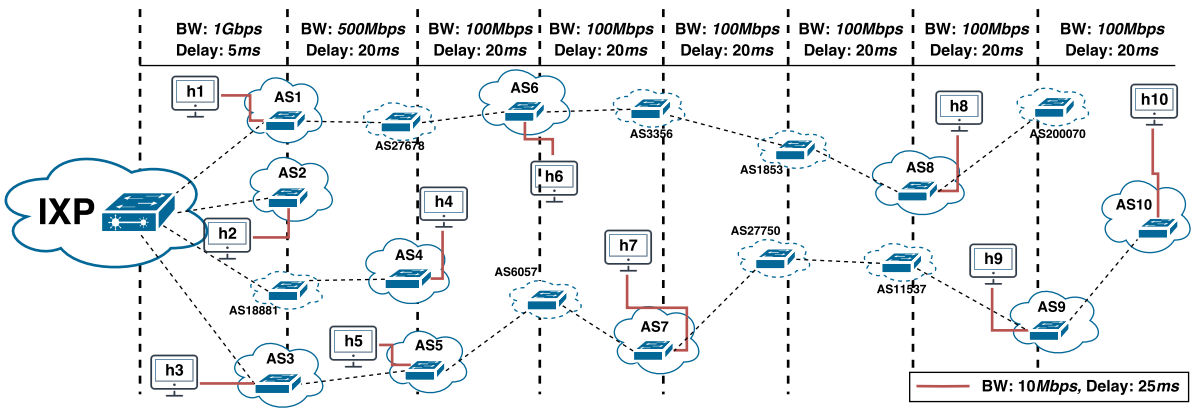


Figure 11 – The IXP-based testing network model.

Table 7 – Cost Map ranking based on the absolute distance between ASes (HopsNumber)

	AS1	**AS2**	**AS3**	**AS4**	**AS5**	**AS6**	**AS7**	**AS8**	**AS9**	**AS10**
AS1	0	2	2	3	3	2	5	5	8	9
AS2	2	0	2	3	3	4	5	7	8	9
AS3	2	2	0	3	1	4	3	7	6	7

Table 8 – Cost Map ranking based on the relative distance between ASes (HopsNumberPTT)

AS	**AS1**	**AS2**	**AS3**	**AS4**	**AS5**	**AS6**	**AS7**	**AS8**	**AS9**	**AS10**
AS1	0	0	0	1	1	2	3	5	6	7
AS2	0	0	0	1	1	2	3	5	6	7
AS3	0	0	0	1	1	2	3	5	6	7

In the case of ALTO information, the same AS-Path Summary file is used in order to build two Cost Maps based on the topological distance expressed as the number of hops between ASes. Each Cost Map ranking's information, both with the absolute and relative distance, can be found in Table 7 and Table 8, respectively. Finally, hosts that belong to ASes present at the IXP (i.e., h1, h2, h3) were defined as ALTO clients, since the BGP data (and topology) is mostly meaningful from the IXP vantage point.

5.3.1.2 Workload and Metrics

For each ALTO client (h1, h2, h3), we run end-to-end round-trip time measurements and available bandwidth with the remaining nine hosts using *ping*[5] and *iperf*[6] tools, respectively. The main idea behind this workload is to emulate a client application (each host) trying to connect with candidate (one of nine possible) peer applications / content servers based on a random selection, and then, to compare the obtained bandwidth and latency if the client had use the ALTO information to perform better-than-random peer selection through the use of the ALTO Cost Map ranking.

We consider both an ideal scenario without traffic as well as another one with a background traffic, using the D-ITG traffic generator — with randomly selected source and destination pairs — sending TCP traffic (512 byte packet size, 1,000 pps rate).

5.3.1.3 Results Analysis

Overall, the results are encouraging as one may expect from applications being able to choose destination peers using ALTO information instead of a random peer selection. Applications with built-in module to evaluate the network performance from/to each candidate peer would correspond to the optimal choice from the application point of view. However, this

[5] Average Round-trip time (ms) of 30 *pings* is computed.
[6] Average TCP throughput (Kbps) in 20 Sec is computed.

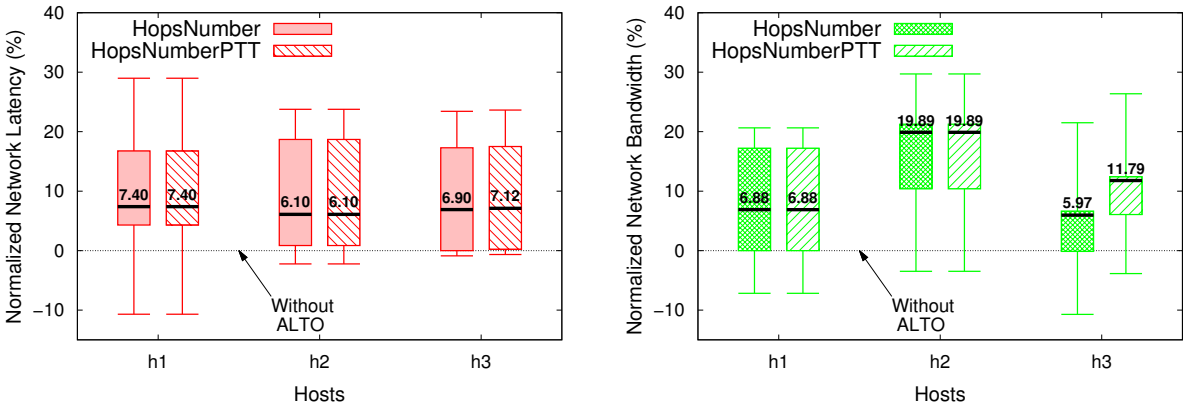


Figure 12 – Network latency and throughput gains (scenario with background traffic) of h1, h2 and h3 using ALTO Cost Map ranking based on an absolute distance (HopsNumber) and relative distance (HopsNumberPTT) metrics compared to random peer selection.

may not be the best one regarding the network operator policies (e.g., avoid transit costs) and certainly not the simplest (extra code needed per application) nor the quickest method (the application needs to assess all candidate destination IPs prior to connection setup).

Figure 12 shows the normalized latency and bandwidth (as box plots with mean, quartiles, and max/min values) that an ALTO client would obtain when using the Cost Map ranking compared to a random selection approach (average values of 8 samples are used as baseline) in a scenario with background traffic. Without traffic, results are also positive and more expressive in terms of latency gains (see Table 9) when compared to throughput improvements.

Results show an improvement in latency (Fig. 12a) and throughput (Fig. 12b) of up to 29%. In some cases, peers selected through AaaS may end up with slower bandwidth or higher latency (between 1 and 11%). This means that while reducing the total number of AS hops could result in significant performance improvements, shortest paths in the Internet are not always the best (e.g., due to congestion). However, this under-performance represents, on average, less than 25 percent of all cases.

A comparison of performance between two proposed maps (Table 7 and Table 8) is also considered. For example, the Cost Map based on absolute distance (HopsNumber) suggests h3 (AS3) to seek out h5 (AS5), while the Cost Map based on relative distance (HopsNumberPTT) informs h3 (AS3) to connect to h1 (AS1). In both cases, performance improvements above 20% (latency and throughput) are obtained, yet when h3 uses the IXP infrastructure to select a peer (h1), as HopsNumberPTT suggests, further throughput improvements (up to 26%) and lower latency (up to 24%) can be obtained.

Table 9 – Network Latency (ms) in a scenario with no traffic expressed as RTT AVG and ±RTT MDEV.

	h1	**h2**	**h3**	**h4**	**h5**	**h6**	**h7**	**h8**	**h9**	**h10**
h1	—	120.5	120.6	160.6	160.6	180.5	240.8	300.7	360.9	401.0
	—	(±0.4)	(±0.3)	(±0.5)	(±0.5)	(±0.7)	(±0.5)	(±0.8)	(±0.8)	(±1.0)
h2	120.5	—	120.4	160.6	160.6	200.7	240.9	320.9	360.9	401.1
	(±0.3)	—	(±0.2)	(±0.4)	(±0.4)	(±0.3)	(±0.6)	(±0.7)	(±0.8)	(±0.6)
h3	120.5	120.5	—	160.5	140.5	200.6	220.6	320.8	340.8	380.8
	(±0.4)	(±0.5)	—	(±0.6)	(±0.4)	(±0.5)	(±0.4)	(±0.5)	(±0.4)	(±0.8)

5.3.2 Inter-Domain Traffic Reduction

This scenario is an extended version of the previous Mininet topology (Fig. 11). The idea now is to emulate one of the most popular P2P file-sharing applications such as BitTorrent[7]. It includes a resource directory (a Tracker) and P2P clients (BitTorrent clients), with an ALTO server that might be subsequently accessed by a BitTorrent client in order to obtain topology information of destination peers (before connecting) and to attempt to reduce the Inter-domain traffic by prioritizing peers within the same AS.

5.3.2.1 Experimental Setup

Figure 13 shows the distribution of new components in our extended network model: (i) a Tracker (placed on AS1853), (ii) BitTorrent clients (all hosts in each AS), (iii) an ALTO server (placed on IXP) and (iv) a host that is sharing a file (h100AS10).

For the Tracker and BitTorrent clients, we re-use a Java implementation of the BitTorrent protocol[8]. It provides two packages:

a) trackerBT: Single and easy-to-use Tracker[9] (HTTP server) for BitTorrent clients. The information of torrent files and peers that are sharing is stored as XML files.

b) jbittorrent (JBIT for short): This BitTorrent client allows the creation of torrent files[10] and the download and sharing of files between peers.

Our ALTO server uses the same topology information on the first experiment (See Experimental Setup, Subsection 5.3.1) and, each of the five hosts in each AS directly connected to the IXP (AS1, AS2, AS3) is an ALTO client. In order to reduce inter-AS traffic, we are considering the Cost Map ranking based on the absolute distance (See Table 7), so that an ALTO

[7] http://www.bittorrent.org/index.html
[8] More information and source code: https://github.com/cloudspaces/jbittorrent
[9] Type of server for facilitating the communications between peers.
[10] Each torrent file contain information about the piece length, file size, Tracker(s) URL(s), and others.

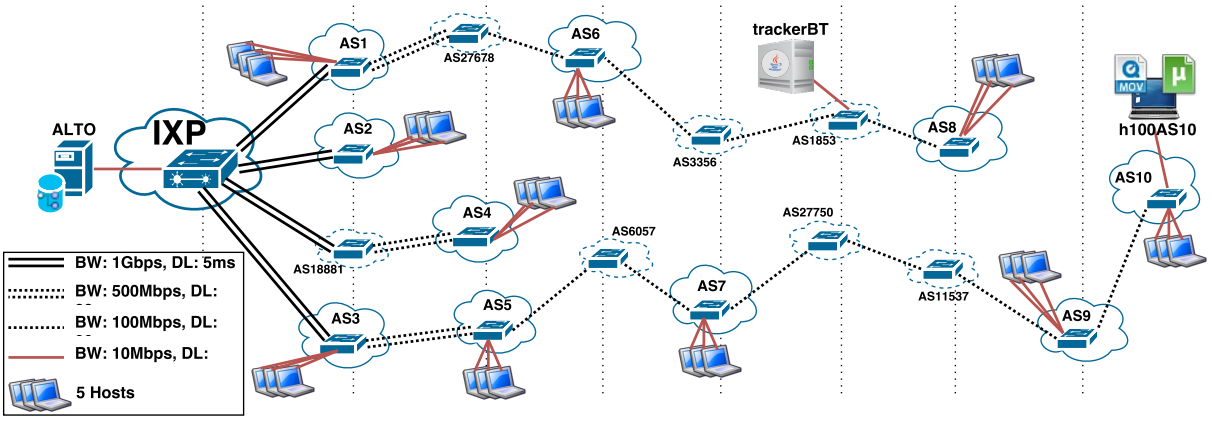


Figure 13 – The IXP-based testing network model (extended version).

client always prefers another local ALTO client (i.e. hosts connected to the same AS/switch) first. In addition, a new package based on the JBIT API was generated (called JBIT_AaaS) to consume the ALTO services.

In order to determine whether AaaS can reduce cross-domain traffic, the performance (in terms of AS-level hops) of the better-than-random peer selection recommended by JBIT_AaaS (referred to as *JBIT_AaaS-recommended peers*) is compared to the random peer selection employed by JBIT (referred to as *JBIT-recommended peers*).

5.3.2.2 Workload and Metrics

A procedural flow for the first experiment (using the JBIT_AaaS API) is shown in Figure 14. The procedure details are explained below:

1. Start the Tracker and the ALTO server.
2. A file of 49 MB (test1.mov) will be shared by h100AS10 (AS10). This peer creates a torrent file (test1.torrent) using BitTorrent protocol with 256 KB of piece length (approximately 197 pieces).
3. Then, the torrent file is published to the Tracker.
4. Copy the test1.torrent file onto each of the other hosts (BitTorrent clients).
5. h100AS10 starts the BitTorrent protocol to share a file (test1.mov). This peer becomes a Seeder[11].
6. A randomly selected source host (SRC peer) uses the torrent file (previously copied) and it starts the BitTorrent protocol:

[11] Status when a peer possesses 100% of the data and it starts uploading content.

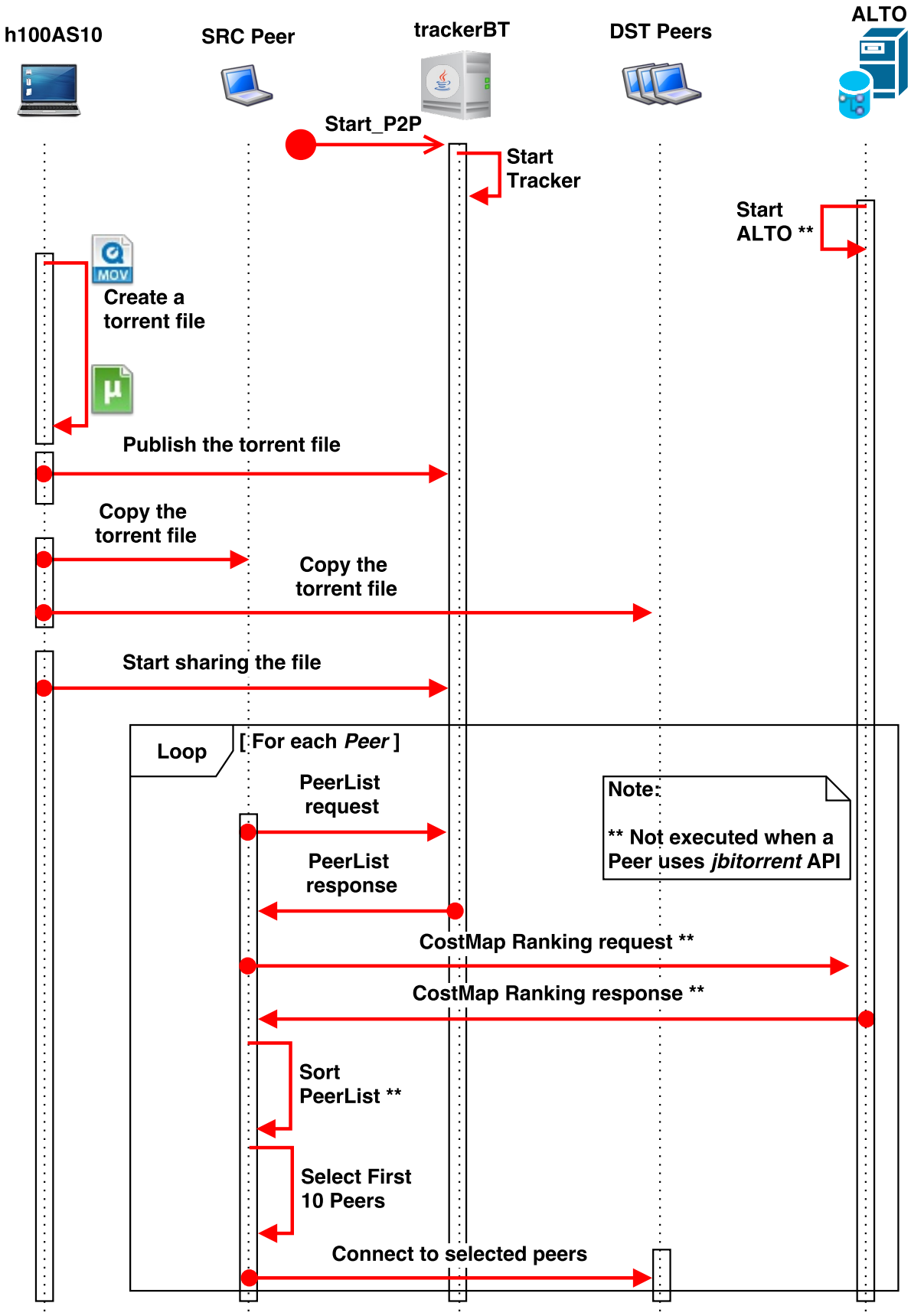


Figure 14 – Flow for the IXP-based testing network model (extended version).

6.1. Using a GET request to the Tracker, a destination peer list (DST peers) is retrieved. Eventually, when all peers in our topology start the BitTorrent protocol, the maximum size of the peer list will be 49.

6.2. Match each peer (SRC and DST) with its corresponding AS number. Afterwards, from the ALTO server, get a Cost Map ranking based on the hops number between ASes (from SRC to DSTs).

6.3. Based on the Cost Map ranking, the original peer list is sorted. Since a lower cost value indicates a higher preference, peers that belong to the same AS (cost: 0) are placed first, followed by peers whose cost:1, and so on.

6.4. For this experiment, the maximum number of connected peers is 10, so that we need to select only the top ten of the peer list.

6.5. Finally, the SRC peer connects to the selected peers and the file download begins. This peer becomes a Leecher[12].

7. Repeat the step 6 for each of the other 49 hosts.

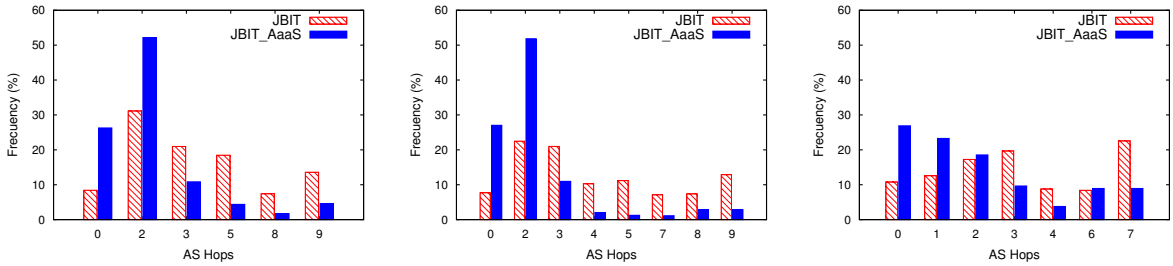
For the second experiment, we are following the standard workflow of the BitTorrent protocol, using the JBIT API. It means that an ALTO server is not considered in this case. We are therefore executing the steps previously described (Fig. 14) without considering: (i) the Cost Map ranking request and response and (ii) the process of peer list sorted. Hence, after a SRC peer retrieves the destination peer list, the first 10 peers are considered — in the order in which they are received by the Tracker.

Both experiments are executed 10 times. For each execution, each SRC peer generates a sample with the number of pieces extracted from the DST peers, in order to download the full content of a file (test1.mov). After that, and since each peer is associated with an ASN, it is possible to compute the amount of AS-level hops to reach destination peers. By doing so, we aim at knowing the number of pieces retrieved in a given number of hops for each SRC peer.

5.3.2.3 Results Analysis

In our analysis, let us just consider the hosts whose ASes are directly connected to IXP (AS1, AS2 and AS3). Figure 15 plots the AS-hops number distribution for each SRC peer grouped by AS. Each value in the *y* axis represents the percentage of downloaded pieces in *x* number of AS-hops. AS-hops value equal to 0 means that both SRC and DST peers belong to the same AS, and thus intra-AS traffic is being generated between them. On the other hand, a higher value indicates which DST peers selection is crossing the network boundaries that is generating Inter-domain traffic.

[12] Status when a peer is downloading a file and does not have the entire file. However, completed pieces can be distributed to other peers that do not have the part it is owning.



(a) AS1 (b) AS2 (c) AS3

Figure 15 – AS-Hops Distribution

Using DST peers selection recommended by JBIT evidenced that the amount of AS-level hops required to download a whole file is highly concentrated from 2 to 5 AS-hops from source AS. For example, in the case of AS1 (Fig. 15a) and AS2 (Fig. 15b), the number of pieces recovered from 2, 3, 4 and 5 AS-hops represents 71 and 65% of the total, respectively. A similar behaviour is exhibited in AS3 (Fig. 15c), showing even the largest number of pieces (23%) from DST peers, with a depth of 7 AS-hops (i.e. peer members of AS8 and AS10). Consequently, the amount of Intra-domain traffic generated reaches only between 7 and 11%. In addition, the average number of AS-level hops for the three ASes varies between 3.5 and 4.3 hops.

Alternatively, selecting DST peers using JBIT_AaaS results in a better performance. In this experiment, the higher concentration of pieces downloaded is at a depth between 0 and 2 AS-hops. In AS1 and AS2, for instance, 78% of the total number of pieces is extracted from the same AS (26%) or they are along 2 AS-hops (52%); whilst for AS3 the same number of AS-hops represents 69%. Moreover, we note that only 6% of the total content in a file is generated by DST peers along paths between 8 and 9 AS-hops deep (higher depths). The average number of AS-hops is also reduced between 1.9 and 2.1 AS-hops.

We also include the accumulated AS-level hops of the three ASes (Fig. 16) in order to illustrate the overall percent of Intra- and Inter-domain traffic generated in our test scenario. For example, when we use an ALTO server (JBIT_AaaS approach), what emerges most clearly is that, over 28% of the times, it selects peers that do not leave the AS source. This represents more than three times the performance achievable through JBIT (9%) at random. This difference is attributed to the fact that JBIT_AaaS provides locality information (using the Cost Map ranking) and, therefore, DST peers along paths within a single AS (Cost: 0) are the ones selected in the first place.

Regarding cross-AS traffic, the figure also shows that DTS peers reached by JBIT_AaaS are close to the SRC peer. For instance, 50% of the total pieces are retrieved at a depth of 1 and 2 AS-hops, while JBIT-recommended peers in the same depth can only download 28% of the blocks for a file. Furthermore, considering the greater depths (AS8, AS9 and AS10), which in our Cost Map ranking show a higher value, i.e. a lower preference (See Table 7), the sum of

pieces retrieved from those ASes through SRC peers running the JBIT_AaaS API (7%) is less than three times the sum of pieces retrieved at random by JBIT (24%).

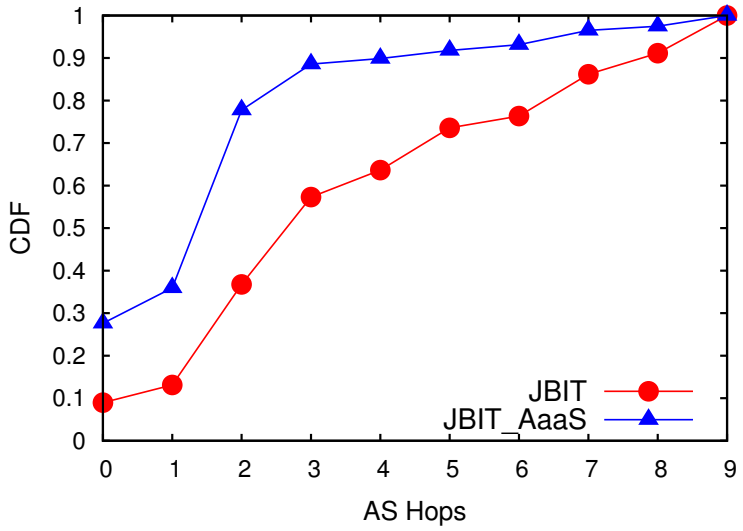


Figure 16 – CDF of number of AS-hops to reach JBIT_AaaS-recommended peers and those from JBIT.

5.3.3 Performance Improvement of a real P2P Application

In this third use case, we present a scenario based on the configuration of the above experiment (See Fig. 13). However, BitTorrent applications now will obtain a real file from large peers distributed on the Internet. The main goal is to provide underline topology information from our public ALTO server in a way that enables BitTorrent client applications to improve the transfer rate performance and, therefore, decrease download time. In addition, we are comparing the performance of our two Cost Maps (HopsNumber and HopsNumberPTT) and their effect on Inter-domain traffic reduction.

5.3.3.1 Experimental Setup

We are following the same sequence diagram shown in Figure 14 in order to download five different open-source applications. However, for each element (SRC peer, DST peers, ALTO server and Tracker) some considerations and/or limitations need to be taken into account:

In this experiment we first obtain a torrent file (containing meta-information such as Piece length, File name, File size, public Tracker URL, etc.) for each application. After that, it is necessary to edit the Tracker URL[13] and to remove UDP Trackers. The reason for such edition is that our Java implementation only supports HTTP GET requests made by the TCP-based Trackers. Table 10 shows more detailed information of the test torrent files, including the number of Seeders and Leechers returned by a Tracker.

[13] A online tool to edit torrent files is used: http://torrenteditor.com/

Table 10 – Torrent Files description (April 19th, 2016)

Name	# Seeder	# Leecher	Piece Length	File Name	File Size
Torrent1.torrent	1446	42	512 KB	ubuntu-15.10-desktop-i386.iso	1.14 GB
Torrent2.torrent	183	6	512 KB	debian-8.4.0-amd64-CD-1.iso	630 MB
Torrent3.torrent	558	22	256 KB	LibreOffice_5.1.2_Win_x86.msi	211.36 MB
Torrent4.torrent	304	35	512 KB	ubuntu-14.10-desktop-i386.iso	1.11 GB
Torrent5.torrent	3576	68	512 KB	ubuntu-15.10-desktop-amd64.iso	1.10 GB

We use a single SRC peer located at the State University of Campinas (ASN: 53187). The BitTorrent protocol is started by SRC peer, either running the JBIT or the JBIT_AaaS API. Regarding the latter, a new input parameter is added in order to obtain the absolute or relative distance between the SRC peer and DST peers. In addition, the maximum number of the DST peer list to be received from a Tracker is 100 and, our SRC peer is considering only IPv4 addresses. Then, the list is sorted (in the case of JBIT_AaaS) and then half of the peers are selected for the final DST peer list.

Regarding the ALTO server, unlike the two previous use cases where we created an alternative server based on simplified topology information, now the DST peer list consists of real hosts on the Internet, so that we are querying our real ALTO server prototype built from real BGP data at 25 Brazilian public IXPs.

5.3.3.2 Workload and Metrics

We are assessing the download rate, the download time and the amount of Inter-AS traffic generated by a BitTorrent client (SRC peer) by running the JBIT API and the JBIT_AaaS API. We also compare the performance variation of the JBIT_AaaS API, using the Cost Map with the absolute distance (HopsNumber) and the relative distance (HopsNumberPTT) between ASes.

For each torrent file, the SRC peer runs the BitTorrent protocol 20 times. JBIT and JBIT_AaaS with the HopsNumber parameter are executed in parallel ten times, while JBIT and JBIT_AaaS with the HopsNumberPTT parameter are executed another ten times (also in parallel). Every 10 seconds, while each application is being downloaded, the download rate (sum of all download rates in each DST peer connected) is computed. Afterwards, it is also possible to compute the total download time at the end of each execution. Finally, we can associate each DST peer with its ASN in order to know the number of ASes that are accessed — generating Inter-domain traffic — to download a specific file.

5.3.3.3 Results Analysis

Based on experimental results, we can state that our approach (AaaS) provides a better-than-random peer selection, enabling a better performance of P2P applications through increased network bandwidth and, consequently, reducing the total download time. In addition, the fact that DST peers are concentrated in a few ASes reduces the amount of inter-AS traffic and therefore the operating cost of an ISP decreases as well.

For each torrent file, Figure 17 shows the download rates (as candlesticks with median, quartiles, and max/min values) from JBIT_AaaS-recommended DST peers and those from DST peers picked at random by JBIT. For this and for Figure 18, we are only using transfer rates greater than the piece length (See column 2 of Table 10). This is because the last pieces arrive more slowly (phenomenon referred to as *last piece problem*[14]) and they do not contribute meaningfully to our results.

When using our Cost Map ranking with the absolute distance, the overall results are very encouraging. For example, as Figure 17a shows, the difference of median download rates for JBIT_AaaS and JBIT is between 166 and 178 KBytes/s, with the exception of torrent2, where the difference is only 30 KBytes/s and torrent3, which is slightly lower (-26 KBytes/s). Another important observation is that DST peers reached by JBIT generally provide higher peak download rate than those selected by JBIT_AaaS (torrent2, torrent4 and torrent5). This behavior is normal and expected because the JBIT API splits each download rate into a large and random number of peers (See column 2 of Table 11), allowing to achieve a high global transfer rate. In the case of JBIT_AaaS, the amount of DST peers decreases (column 7 of Table 11) because they are selected based on the Cost Map ranking, i.e. it is possible to connect to the same set of

[14] To complete a file, many peers are missing only a few pieces, which are same for all peers. Usually, if a peer is lucky enough to retrieve the missing pieces, it departs immediately (HAJEK; ZHU, 2010).

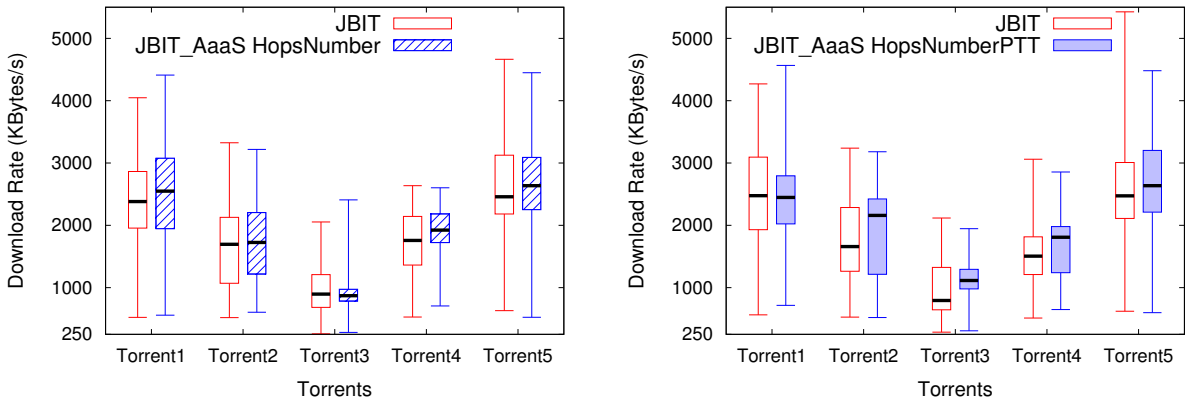


Figure 17 – Global download rates obtained from peers recommended at random by JBIT and those from peers picked by JBIT_AaaS using the Cost Map ranking with the absolute distance (a) and the Cost Map ranking with the relative distance (b).

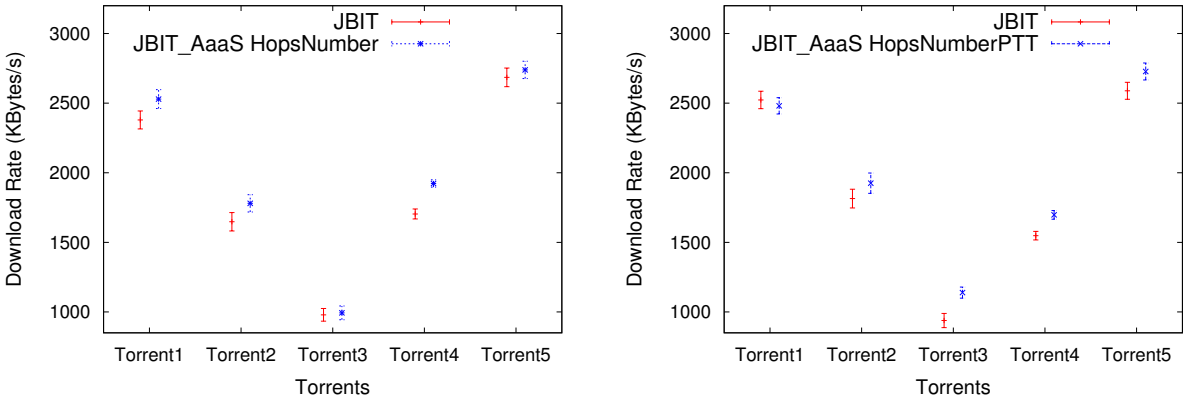


Figure 18 – The mean global download rate at 95% of confidence level obtained from peers located at random by the BitTorrent protocol (JBIT) and those from peers selected by JBIT_AaaS using the Cost Map ranking with the absolute distance (a) and the Cost Map ranking with the relative distance (b).

peers continuously.

A better performance is observed when we use JBIT_AaaS with the Cost Map ranking based on the relative distance (Figure 17b). The difference in median download rate is above 300 KBytes/s (torrent3 and torrent4), reaching around 500 KBytes/s in the case of torrent2. A minor difference is observed when torrent5 file is used (163 Kbytes/s) and only in the case of torrent1 a slight median download rate difference (-29 KBytes/s) is observed. Finally, just like in the first experiment, JBIT achieves the highest global download rates (except with torrent1), and even shows the higher peak download rate of all our experiments (5426 KBytes/s).

Now, we will evaluate whether the performance obtained by JBIT and JBIT_AaaS are significantly different. Figure 18 plots the Confidence Interval (CI) for the mean of download rates with 95% Confidence Level (CL). For example, through an approximate visual test between the CI generated by JBIT and JBIT_AaaS using the Cost Map ranking with the absolute distance (Fig. 18a), it can be noted that CIs for torrent1, torrent2 and torrent4 are not overlapping, i.e. the upper edge of the lower CI (JBIT) is bellow the lower edge of the upper CI (JBIT_AaaS). Thus, we can state at 95% of confidence that peers selected by JBIT_AaaS (hopsNumber) have a higher download rate than peers picked by JBIT at random. The chance of error in this statement is 5%. In the case of torrent3 and torrent5, CIs are overlapped and the mean of JBIT_AaaS is in the CI of JBIT. Therefore, it is not possible to state that JBIT_AaaS has a better performance than JBIT or vice versa.

JBIT_AaaS with the relative distance (HopsNumberPTT) shows, again, better download rates when compared to JBIT (See Fig. 12). As we can see in torrent3, torrent4 and torrent5, the CI obtained by JBIT is above the CI generated by JBIT_AaaS using our ALTO information. Thus, our approach shows statistically significantly better download rate than a standard Bit-

Table 11 – Statistics obtained from JBIT-recommended peers and those from peers picked by JBIT_AaaS using the Cost Map ranking with the absolute distance.

Torrent File	JBIT					JBIT_AaaS (HopsNumber)				
	Peers			ASes		Peers			ASes	
	Nº	Download★ Rate (KB/s)	R^{+}	Nº	Depth*	Nº	Download★ Rate (KB/s)	R^{+}	Nº	Depth*
Torrent1	221	98.9 (±7.4)	0.66	125	3.7 [3.2]	166	96.4 (±7.6)	0.63	75	3.6 [2.9]
Torrent2	74	99.6 (±13.9)	0.64	50	3.7 [3.1]	48	102.8 (±16.9)	0.74	28	3.5 [2.8]
Torrent3	116	72.6 (±14.1)	0.90	90	3.9 [3.3]	77	88.2 (±19.1)	0.68	54	3.5 [2.8]
Torrent4	54	108.9 (±18.6)	0.72	41	3.7 [3.2]	30	116.2 (±22.8)	0.86	23	3.5 [2.7]
Torrent5	281	104.9 (±6.7)	0.66	161	3.8 [3.3]	230	112.9 (±16.8)	0.86	105	3.5 [2.8]

★ *Mean download rate at 95% of CL.*

✛ *Correlation coefficient (R) between mean download rate and percent of downloaded pieces per peer.*

✻ *Mean AS-level hops computed by Traceroute and our ALTO Cost Map (relative distance).*

Torrent protocol. In torrent2, it is necessary to apply a *t*-test[15] due to the CIs overlap, but mean of one is not in the CI of the other. Once the *t*-test has been calculated, we observe that the CI for mean difference at 95% CL does not include zero, which means that JBIT_AaaS is again better than JBIT. Finally, when the results of torrent1 are compared, CIs overlap and mean of one (JBIT) is in the CI of the other (JBIT_AaaS), i.e. both alternatives are not significantly different.

As we already mentioned, the BitTorrent protocol gets the highest global download rates by splitting its bandwidth in a large number of DST peers. However, this usually means that each connection obtains a relatively low individual transfer rate. For example, Table 11 shows (among other values) the mean download rate at 95% Confidence Level (CL) for all peers generated by JBIT (3rd column) and JBIT_AaaS (8th column). As we can see, overall, the mean download rate reached by JBIT_AaaS-recommended peers is higher than those peers selected at random by JBIT. This suggests that pieces of a file are downloaded faster using the JBIT_AaaS API and, consequently, decreases the total download time. To verify this supposition, the number of pieces retrieved for each DST peer is computed (expressed as a percent). Subsequently, we calculate the Correlation Coefficient (CC) between the download rate and the percent of downloaded pieces per DST peer. The main idea is to know whether our SRC peer obtains the greatest number of pieces from DST peers with high download rates.

Table 11 shows, for each torrent file, the CC obtained by JBIT (4th column) and JBIT_AaaS with HopsNumber (9th column). On average, JBIT_AaaS obtains a CC value greater than peers randomly selected by JBIT. To give a few examples, the value of CC for torrent4 and torrent5 using JBIT_AaaS is 0.86. This is considered a strong positive correlation, which means

[15] A t-test is used to know if two groups of data are significantly different from each other.

Table 12 – Statistics obtained from JBIT-recommended peers and those from peers picked by JBIT_AaaS using the Cost Map ranking with the relative distance.

Torrent File	JBIT					JBIT_AaaS (HopsNumberPTT)				
	Peers			ASes		Peers			ASes	
	Nº	Download★ Rate (KB/s)	R^{+}	Nº	Depth*	Nº	Download★ Rate (KB/s)	R^{+}	Nº	Depth*
Torrent1	199	104.7 (±8.0)	0.64	113	3.6 [3.2]	157	96.8 (±8.0)	0.67	68	3.6 [3.0]
Torrent2	75	103.4 (±13.1)	0.60	48	3.6 [3.1]	50	94.1 (±18.0)	0.82	33	3.6 [2.9]
Torrent3	112	79.9 (±14.1)	0.73	80	3.8 [3.3]	78	86.6 (±10.9)	0.65	55	3.6 [2.9]
Torrent4	59	101.8 (±18.0)	0.70	45	3.6 [3.2]	37	108.5 (±20.3)	0.80	23	3.4 [2.8]
Torrent5	268	99.9 (±7.3)	0.56	159	3.7 [3.1]	246	101.4 (±7.7)	0.69	106	3.5 [3.0]

★ *Mean download rate at 95% of CL.*

✢ *Correlation coefficient (R) between mean download rate and percent of downloaded pieces per peer.*

✻ *Mean AS-level hops computed by Traceroute and our ALTO Cost Map (relative distance).*

that high download rate scores go with high percent of downloaded pieces (and vice versa). The best CC value is reached by JBIT in torrent3 (0.9), however, in others torrent files, the CC varies between 0.66 and 0.72, values that are considered moderate positive correlations.

A comparison between the CC generated by JBIT and JBIT_AaaS (HopsNumberPTT) is presented in Table 12. In this experiment, once again a better performance is achieved through peers found by JBIT_AaaS than through those picked at random. By comparing the CC values, it is only in the case of torrent3 when JBIT (0.73) reaches a slightly larger CC value than JBIT_AaaS (0.65). Furthermore, three moderate positive correlations (torrent1, torrent3 and torrent5) and two strong positive correlations (torrent2 and torrent4) are obtained using an ALTO server, while there are only CC values with moderate positive correlations in the case of JBIT.

It is also possible to determine whether our approach may be helpful for reducing Inter-AS traffic. To do so, the ASN is assigned to each DST peer, so we can know the amount of traffic that crosses network boundaries to download content. Table 11 and Table 12 display information about how many ASes have been accessed by JBIT (5th column) and JBIT_AaaS (10th column). Just as with the amount of peers, the number of ASes reached by JBIT_AaaS proves to be lower than the number of ASes selected by a standard BitTorrent Protocol (JBIT). For example, using our Cost Map ranking with the absolute distance (HopsNumber), on average, the number of ASes decreases 41%. We also noted a substantial reduction using the other Cost Map (HopsNumberPTT) at the rate of 37%. This clearly indicates that our approach allows a higher concentration of DST peers that belong to a same AS and consequently there is a decrease in the total amount of Inter-domain traffic generated, since it does not constantly cross the network boundaries.

Additionally, with the help of the *traceroute* tool, which gives us information about

the routes that packets take at the IP level (data plane) and grouping these sequence of IP addresses by AS, it is possible to calculate the amount of AS-level hops to reach JBIT_AaaS-recommended peers and those picked by JBIT at random[16]. In table 11, we can compare the average number of hops for DST peers located by JBIT (6th column) and JBIT_AaaS with HopsNumber (11th column). On average, the observed depth of a BitTorrent client is 3.8 when it is using a randomly peer selection. Alternatively, when locality information of peers is used, depth values lie between 3.5 and 3.6. In the other experiment, JBIT_AaaS (considering the Cost Map with the relative distance) still maintains a better performance than JBIT (see Table 12). For example, the average AS-level hops to reach a DST peer is between 3.4 and 3.6 AS-level hops using AaaS, while for peers selected by JBIT, the average depth is between 3.6 and 3.8 AS-hops. Both results indicate that the Internet content of a BitTorrent client could be obtained from the closest peers using topology information provided by AaaS. This translates in cost reduction of Inter-domain traffic as well as in an enhanced user experience (e.g. faster access to content).

In the same columns (6th and 11th) in square brackets, we present additional information to show the AS-level hops obtained from the Cost Map in our ALTO server. An aspect to be highlighted is the fact that the average amount of hops is lower compared with the AS-hops achieved by *traceroute*. While the default behavior of BGP is to prefer shorter routes (just like our ALTO Cost Map), what would explain this difference is the practice of traffic engineering techniques (for example, based on AS-Prepend), which are used for selecting a less costly path or a path with fastest connection, and not necessarily the shortest AS Path. However, as with *traceroute*, DST peers selected by JBIT_AaaS have lower AS-hops than those selected by JBIT (on average, 2.9 and 3.2 respectively). It means that our Cost Maps that provide AS-level hops (created from control plane BGP information) are comparable with the AS-hops along the data plane, making AaaS a reliable service to select the best destination in terms of topological distance.

Finally, we evaluate the effects on the download time when a BitTorrent client uses the services provided by our ALTO server. Figure 19 presents the total time (in seconds) required by JBIT and JBIT_AaaS for download completion. It should be pointed out that we are only considering download times up to 99.5% of the total due to a phenomenon known as *last piece problem* (described above).

In Figure 19a we can see a comparison between the time it takes to download a file through JBIT and JBIT_AaaS (HopsNumber). As was to be expected, the maximum values (i.e. longer download time) were obtained by JBIT, except for torrent5. In this case, the maximum download rates were also generated by JBIT (See Fig. 17a), what would explain this behavior. However, the difference of median values are generally in favour of JBIT_AaaS, thereby indi-

[16] It should be pointed out that in our measurement, we are considering approximately 65% of the total destination peers, because many of them were unreachable, once it was impossible to trace their path while the Traceroute tool was running.

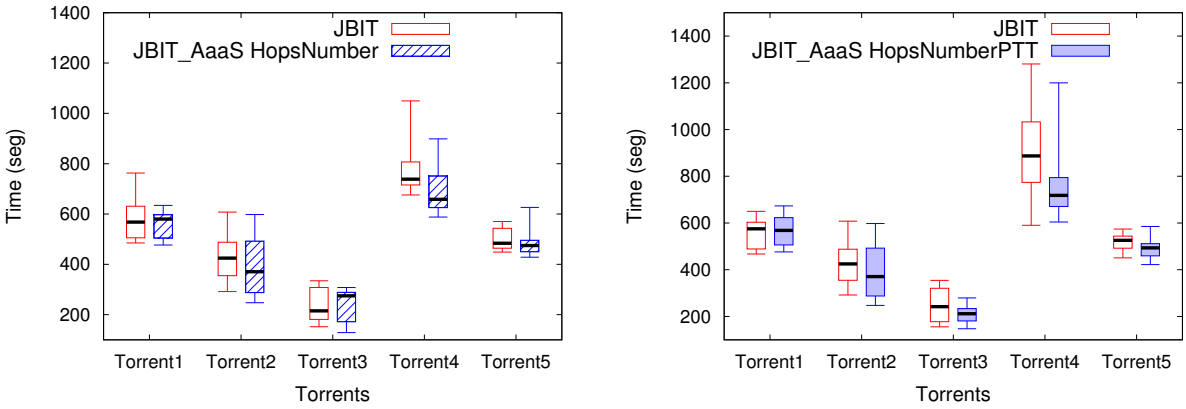


Figure 19 – Total download times (as candlesticks with median, quartiles, and max/min values as the 95-/5-percentiles) obtained by JBIT vs. JBIT_AaaS using the Cost Map ranking with the absolute distance (a) and JBIT vs. JBIT_AaaS using the Cost Map ranking with the relative distance (b).

cating a shorter download time. The greatest median difference is calculated in torrent4 (-80 seg.), what makes sense considering the fact that JBIT also achieves the best mean download rate difference with torrent4 (See Fig. 18a).

A similar behaviour is seen when we compare JBIT and JBIT_AaaS using the Cost Map with the relative distance (Fig. 19b). For all torrent files, the median difference indicates that the time to complete a download is reduced when an application uses topological information provided by AaaS. Even torrent4 achieves the best difference (-169 seg.), which, in fact, is in line with our expectations, since this file generated one of the highest mean download rate values and correlation coefficient per peer (see Table 12).

6 Conclusions and Future Work

In the past few years, the amount of Internet traffic generated by distributed applications (File sharing, CDNs, Live media streaming, Real Time Communications, etc.) has enormously grown and led to the increasing use of network resources. Likewise, a large amount of data is transferred through connections between hosts/servers distributed across the Internet, while demonstrating a limited knowledge of the underlying network topology. In order to provide a solution for these and other challenges, we presented the recently standardized IETF ALTO protocol (RFC 7285), that constitutes a service for network applications so that the best resource (according to established policies) can be intelligently selected. Based on this protocol, we also presented the design of a public service, referred to as ALTO-as-a-Service (AaaS), with the potential of benefiting both network operators and applications.

To the best of our knowledge, this is the first work that explores the use of inter-domain routing data publicly available at IXPs to create abstract topology and cost maps following the ALTO protocol. Our architecture encompasses the whole process of ALTO service delivery, from the BGP raw data extraction, its subsequent formatting, filtering, and assembling, throughout its transformation into ALTO information, through its persistent storage, and finally the creation of a public ALTO server. Therefore, it makes it possible to prove, that not just network operators, but also third parties or user communities can provide ALTO services from public network information.

In order to consolidate and put into practice our design (AaaS), we implemented a proof-of-concept. In this first version, the input used was the BGP routing information at 25 Brazilian public IXPs (resulting in over 2.5 GB of data). In addition, the popular Neo4j graph database and the OpenDaylight controller were selected as ALTO server Back-end and Front-end, respectively. Afterwards, the network map and cost map were assessed, both in terms of functional behaviour (in accordance with the ALTO specifications) and of performance profiling (system response time).

At the same time, we validated our proposal through three use case scenarios, using a IXP-based testing network model and a real P2P testing environment with real Trackers, and establishing connections between destination peers around the world. Based on our analysis, our work demonstrates the potential applications (e.g., P2P clients or trackers, CDNs, SDN controllers) to leverage the network awareness provided by ALTO servers, so as to optimize their decision-making processes regarding endpoints selection and, consequently, providing a better user experience. These decisions based on the topological distance demonstrate the importance of minimizing the total number of AS-level hops, and, although we understand it is not the only factor, it could significantly enhance the performance of Internet applications. Finally,

we also showed that our approach can reduce the amount of traffic that crosses network boundaries, resulting in ISPs' cost reductions. Furthermore, in the case of ALTO service providers (in our case IX.br operators), they can benefit from increased, localized IXP traffic exchange.

Our win-win proposal is not free of limitations and there is a challenging amount of future work to be undertaken. Firstly, the AaaS workflow does not handle periodic updates (e.g. when a new dataset is retrieved) and a single snapshot (Dec. 2014) was used in order to built the ALTO information. As a result, the Network Map could not consider some ASNs (PIDs) and potential destination peers (Endpoints). In addition, since information about AS links is not updated in our AS-level graph, the Cost Map service is not be able to select shorter and/or alternative AS-Paths between two PIDs. Therefore, the creation of a procedural flow where the AaaS can support scenarios with multiple datasets is at the top in our future research activities.

The ALTO protocol specification allows that the ALTO information be updated dynamically based on network conditions. However, the Cost Map rankings provided by AaaS are based only on relatively static AS-Path distance and do not consider more dynamic information such as actual bandwidth, latency, packet loss rate, etc. Thus, dynamic updates of cost maps based on public Internet quality measurements (e.g., SIMET Traffic Measurement System, RIPE TTM) are also included in our research agenda.

Resource saturation may also be of particular concern. For example, what if a large number of clients chooses to use a same destination endpoint because it is cheaper? Solutions to this potential issue are being considered as future work activities, such as ALTO servers replying with round robin chosen versions of the maps for equal cost paths to reduce the chances of multiple clients choosing the same best destinations at once.

With the introduction of SDN, it is possible to consider new architectures on the ALTO design. In a vertical architecture (XIE *et al.*, 2012), for example, a network has one or multiple SDN domains, each controlled by an individual SDN controller and a single ALTO server. In this way, it would be possible to create ALTO information from network information provided by SDN Controller (as ALTO clients) and complement it with the network information from others resources (Routing information at IXP, public broadband data, etc.). Moreover, an ALTO server may eventually exchange network information with other ALTO servers in the same or in another the administrative domain. For these reasons, the intersection of ALTO with SDN-controlled domains is also an avenue of ongoing investigation as a means to facilitate inter-domain traffic engineering and SDN east/west interfaces.

Implementing the remaining ALTO services, such as the full Map-Service, the Endpoint Property Service (EPS) and the Endpoint Cost Service (ECS) are other future open tasks in our prototype, which adds to a number of performance optimization opportunities, most of them related to security (e.g. authenticity and confidentiality of ALTO information, availability of ALTO services, privacy of ALTO clients) and Neo4j query tuning techniques (e.g., DB indexes, heap size, garbage collection and Linux fs configuration).

Bibliography

ALIMI, R.; PENNO, R.; YANG, Y.; KIESEL, S.; PREVIDI, S.; ROOME, W.; SHALUNOV, S.; WOUNDY, R. *Application-Layer Traffic Optimization (ALTO) Protocol*. [S.l.], 2014.

BONAVENTURE, O.; SAUCEZ, D.; DONNET, B. *The case for an informed path selection service*. [S.l.], 2008.

BRAY, T. *The JavaScript Object Notation (JSON) Data Interchange Format*. [S.l.], 2014.

BRITO, S.; SANTOS, M.; FONTES, R.; LACHOS, D.; ROTHENBERG, C. Anatomia do ecossistema de pontos de troca de tráfego publicos na internet do brasil. May 2015. In XXXIII Simpósio Brasileiro de Redes de Computadores (SBRC). Vitória, ES, Brazil.

CHATZIS, N.; SMARAGDAKIS, G.; FELDMANN, A. On the importance of internet exchange points for today's internet ecosystem. *arXiv preprint arXiv:1307.5264*, 2013.

CHOFFNES, D. R.; BUSTAMANTE, F. E. Taming the torrent: a practical approach to reducing cross-isp traffic in peer-to-peer systems. In: ACM. *ACM SIGCOMM Computer Communication Review*. [S.l.], 2008. v. 38, n. 4, p. 363–374.

EUROPEAN Internet Exchange Association 2014 Report on European IXPs. 2014. Disponível em: <https://www.euro-ix.net/m/cms_page_media/82/euro-ix-ixp-report-2014-FINAL_U0r0QDx.pdf>.

FIELDING, R. T. *Architectural styles and the design of network-based software architectures*. Tese (Doutorado) — University of California, Irvine, 2000.

GROSSMAN, D. *New Terminology and Clarifications for Diffserv*. [S.l.], 2002. <http://www.rfc-editor.org/rfc/rfc3260.txt>. Disponível em: <http://www.rfc-editor.org/rfc/rfc3260.txt>.

GUANXIU, L.; SUQI, Y.; XINLI, H. A Novel ALTO Scheme for BitTorrent-Like P2P File Sharing Systems. In: *Proceedings of the 2013 Third International Conference on Intelligent System Design and Engineering Applications*. [S.l.: s.n.], 2013.

GUO, H. *Interoperability Testing of the Application-Layer Traffic Optimization (ALTO) Protocol*. [S.l.], 2015. <http://www.ietf.org/internet-drafts/draft-guo-alto-interop-01.txt>. Disponível em: <http://www.ietf.org/internet-drafts/draft-guo-alto-interop-01.txt>.

GURBANI, V.; GOERGEN, D.; STATE, R.; ENGEL, T. Making historical connections: Building application layer traffic optimization (alto) network and cost maps from public broadband data. In: *Network and Service Management (CNSM), 2014 10th International Conference on*. [S.l.: s.n.], 2014.

HAJEK, B.; ZHU, J. The missing piece syndrome in peer-to-peer communication. In: IEEE. *Information Theory Proceedings (ISIT), 2010 IEEE International Symposium on*. [S.l.], 2010. p. 1748–1752.

INDEX, C. V. N. *Forecast and Methodology, 2014-2019 White Paper*. [S.l.], 2015.

KHAN, A.; KWON, T.; KIM, H.-c.; CHOI, Y. As-level topology collection through looking glass servers. In: *Proceedings of the 2013 Conference on Internet Measurement Conference*. New York, NY, USA: ACM, 2013. (IMC '13), p. 235–242. ISBN 978-1-4503-1953-9. Disponível em: <http://doi.acm.org/10.1145/2504730.2504758>.

MADHUKAR, A.; WILLIAMSON, C. A longitudinal study of p2p traffic classification. In: *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS)*. [S.l.: s.n.], 2006. p. 179–188.

MADHYASTHA, H. V.; ISDAL, T.; PIATEK, M.; DIXON, C.; ANDERSON, T.; KRISHNAMURTHY, A.; VENKATARAMANI, A. iplane: An information plane for distributed services. In: USENIX ASSOCIATION. *Proceedings of the 7th symposium on Operating systems design and implementation*. [S.l.], 2006. p. 367–380.

MONTRESOR, A.; JELASITY, M. Peersim: A scalable p2p simulator. In: IEEE. *Peer-to-Peer Computing, 2009. P2P'09. IEEE Ninth International Conference on*. [S.l.], 2009. p. 99–100.

NEO4J. *The Neo4j Manual v2.3.0-M03*. 2015. Http://neo4j.com/docs/milestone/. Disponível em: <http://neo4j.com/docs/milestone/>.

NORTON, W. B. *The Internet Peering Playbook: Connecting to the Core of the Internet*. [S.l.]: DrPeering Press. USA., 2014.

PINTHONG, N.; LILAKIATSAKUN, W. Performance of bittorrent-like p2p file sharing systems inspired by alto. In: *TENCON IEEE Region 10 Conference (31194)*. [S.l.: s.n.], 2013.

ROSEN, E.; VISWANATHAN, A.; CALLON, R. *Multiprotocol Label Switching Architecture*. [S.l.], 2001. <http://www.rfc-editor.org/rfc/rfc3031.txt>. Disponível em: <http://www.rfc-editor.org/rfc/rfc3031.txt>.

SCHARF, M.; VOITH, T.; ROOME, W.; GAGLIANELLO, B.; STEINER, M.; HILT, V.; GURBANI, V. K. Monitoring and abstraction for networked clouds. In: IEEE. *Intelligence in Next Generation Networks (ICIN)*. [S.l.], 2012. p. 80–85.

SEEDORF, J.; BURGER, E. *Application-Layer Traffic Optimization (ALTO) Problem Statement*. [S.l.], 2009.

SHIBUYA, M.; HEI, Y.; OGISHI, T. ISP-friendly peer selection mechanism with ALTO-like server. In: *Network Operations and Management Symposium (APNOMS)*. [S.l.: s.n.], 2011.

STIEMERLING, M.; KIESEL, S.; SCHARF, M.; SEIDEL, H.; PREVIDI, S. *ALTO Deployment Considerations*. [S.l.], 2016. <http://www.ietf.org/internet-drafts/draft-ietf-alto-deployments-14.txt>. Disponível em: <http://www.ietf.org/internet-drafts/draft-ietf-alto-deployments-14.txt>.

XIE, H.; TSOU, T.; LOPEZ, D.; YIN, H. *Use Cases for ALTO with Software Defined Networks*. [S.l.], 2012. <http://www.ietf.org/internet-drafts/draft-xie-alto-sdn-use-cases-01.txt>. Disponível em: <http://www.ietf.org/internet-drafts/draft-xie-alto-sdn-use-cases-01.txt>.

XIE, H.; YANG, Y. R.; KRISHNAMURTHY, A.; LIU, Y. G.; SILBERSCHATZ, A. P4p: Provider portal for applications. In: ACM. *ACM SIGCOMM Computer Communication Review*. [S.l.], 2008. v. 38, n. 4, p. 351–362.

YANG, W.; ABU-GHAZALEH, N. Gps: a general peer-to-peer simulator and its use for modeling bittorrent. In: IEEE. *Modeling, Analysis, and Simulation of Computer and Telecommunication Systems, 2005. 13th IEEE International Symposium on*. [S.l.], 2005. p. 425–432.

Appendix

APPENDIX A – Publications

- **LACHOS**, D. A. ; BRITO, S. H. B. ; FONTES, R. R. ; ROTHENBERG, C. E. . Delivering Application-Layer Traffic Optimization Services based on Public Routing Data at Internet eXchange Points. In: *XXXIV Simpósio Brasileiros de Redes de Computadores SBRC 2016*, Salvador, BA, Brazil, Jun 2016.
- BRITO, S. H. B.; SANTOS, M. A. S. ; FONTES, R. R. ; **LACHOS, D. A.** ; ROTHENBERG, C. E. . Dissecting the Largest National Ecosystem of Public Internet eXchange Points in Brazil. In: *Passive and Active Measurements PAM 2016*, Hereklion, Crete, Greece, 2016.
- BRITO, S. H. B.; SANTOS, M. A. S. ; FONTES, R. R. ; **LACHOS, D. A.** ; ROTHENBERG, C. E. . Anatomia do Ecossistema de Pontos de Troca de Tráfego Públicos na Internet do Brasil. In: *XXXIII Simpósio Brasileiros de Redes de Computadores SBRC 2015*, Vitória, ES, Brazil, May 2015.
- **LACHOS, D.A.**; ROTHENBERG, C. E. . Delivering Application-Layer Traffic Optimization Services based on Public Routing Information at Internet eXchange Points. In: *VIII Encontro dos Alunos e Docentes do Departamento de Engenharia de Computação e Automação Industrial 2015*, Campinas, SP, Brazil, Set 2015.