



Universidade do Minho

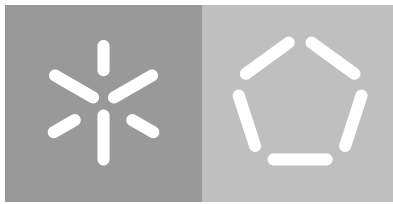
Escola de Engenharia

Departamento de Informática

Paulo Edgar Mendes Caldas

**Development of a system
compliant with the Application-layer
Traffic Optimization protocol**

January 2021



Universidade do Minho

Escola de Engenharia

Departamento de Informática

Paulo Edgar Mendes Caldas

**Development of a system
compliant with the Application-layer
Traffic Optimization protocol**

Masters dissertation

Integrated Master's in Informatics Engineering

Dissertation supervised by

Pedro Nuno Miranda de Sousa

January 2021

AUTHOR COPYRIGHTS AND TERMS OF USAGE BY THIRD PARTIES

This is an academic work which can be utilized by third parties given the compliance of the rules and good practices regarding author and related copyrights, which are internationally accepted.

Therefore, the present work can be utilized according to the terms provided in the license shown below.

If the user needs permission to use the work in conditions not foreseen by the licensing indicated, the user should contact the author, through the RepositóriUM of University of Minho.

License provided to the users of this work



Attribution-NonCommercial-ShareAlike

CC BY-NC-SA

<https://creativecommons.org/licenses/by-nc-sa/4.0/>

[This license allows others to remix, transform and build upon this work, to non-commercial purposes, as long as appropriate credit is given, and the new contributions are licensed under the same license as the original]

ACKNOWLEDGEMENTS

I would like to firstly thank my advisor, professor Pedro Nuno Sousa, who was always present in any moment I struggled and required input to improve on my work.

I would also like to thank my family for financially and emotionally supporting me through my academic journey, the friends I've made along the way that made me see the best in people, and last but not least my dog Oscar who showed me unconditional love like only a dog could.

I finally also thank you, the reader - as a work unused is no work at all, may you find some value in it.

STATEMENT OF INTEGRITY

I hereby declare having conducted this academic work with integrity. I confirm that I have not used plagiarism or any form of undue use of information or falsification of results along the process leading to its elaboration.

I further declare that I have fully acknowledged the Code of Ethical Conduct of the University of Minho.

Paulo Edgar Mendes Caldas

ABSTRACT

With the ever-increasing Internet usage that is following the start of the new decade, the need to optimize this world-scale network of computers becomes a big priority in the technological sphere that has the number of users increasing, as are the *Quality of Service (QoS)* demands by applications in domains such as media streaming or virtual reality.

In the face of rising traffic and stricter application demands, a better understanding of how *Internet Service Providers (ISPs)* should manage their assets is needed. As an effort to optimize the Internet, one important concern is how applications utilize the underlying network infrastructure over which they reside. An evident issue is that most of these applications act with little regard for ISP preferences, as can be evidenced by their lack of care in achieving network proximity among neighboring peers, a feature that would be preferable by network administrators and that could also improve application performance. However, even a best-effort attempt by applications to cooperate will hardly succeed if ISP policies aren't clearly communicated to them. A system to bridge layer interests has thus much potential in helping achieve a mutually beneficial scenario.

The main focus of this thesis is the *Application-Layer Traffic Optimization (ALTO)* working group, which was formed by the *Internet Engineering Task Force (IETF)* to explore standardizations for network state retrieval. The working group devised a request-response protocol where authoritative and trustworthy entities provide guidance to applications in the form of network status information and administrative preferences, with the intent of achieving layer cooperation during normal application operations as a means to reach better Internet efficiency through the optimization of infrastructural resourcefulness and consequential minimization of its operational costs. This work aims to implement and extend upon the ideas of the ALTO working group, as well as verify the developed system's efficiency in a simulated environment.

Keywords: Application-Layer Traffic Optimization, Content Distribution Networks, Network Optimization, Peer-to-Peer, Traffic Engineering

RESUMO

Com o uso cada vez mais acrescido da Internet que acompanha o início da nova década, a necessidade de otimizar esta rede global de computadores passa a ser uma grande prioridade na esfera tecnológica, que vê o seu número de utilizadores a aumentar, assim como a exigência, por parte das aplicações, de novos padrões de Qualidade de Serviço (QoS), como se vê em domínios de stream multimédia em tempo real ou realidade virtual.

Face ao aumento de tráfego e a padrões de exigência aplicacionais mais restritos, uma melhor compreensão é necessária de como os fornecedores de serviços Internet (ISPs) devem gerir os seus recursos. Numa tentativa por otimizar a Internet, um ponto fulcral é o de perceber como as aplicações utilizam os recursos da rede sobre a qual residem. Um problema aparente é a falta de consideração que estas e outras aplicações têm pelas preferências dos ISPs durante a sua operação, como as aplicações P2P pela sua falta de esforço em obter proximidade topológica com os vizinhos na rede overlay, que caso existisse seria preferível por administradores de rede e teria potencial para melhorar o desempenho aplicacional. Todavia, uma tentativa de melhor esforço por parte das aplicações por cooperar não será bem-sucedida se tais preferências não são claramente comunicadas. Um sistema que sirva de ponte de comunicação entre as duas camadas tem portanto bastante potencial na tarefa de atingir um cenário mutuamente benéfico.

O foco principal desta tese é o grupo de trabalho ALTO, que foi formado pelo IETF para explorar standardizações para recolha de informação do estado da rede. Este grupo de trabalho especificou um protocolo de pedido e fornecimento de recursos onde entidades autoritárias auxiliam aplicações com informação sobre estado de rede e preferências administrativas, como forma de obter cooperação entre camadas durante operação aplicacional, para melhor otimizar a Internet através de uma mais eficiente utilização de recursos infraestruturais e a consequente minimização de custos operacionais. Este trabalho pretende implementar e alargar as ideias do grupo ALTO, bem como verificar a eficiência do sistema desenvolvido num ambiente simulado.

Palavras-Chave: Application-Layer Traffic Optimization, Content Distribution networks, Engenharia de Tráfego, Otimização de rede, Peer-to-peer

CONTENTS

Acknowledgements	iii
Abstract	vii
Resumo	ix
List of Figures	xv
List of Tables	xvii
List of Acronyms	xviii
1 INTRODUCTION	1
1.1 Context and motivation	1
1.2 Objectives	4
1.3 Contributions	5
1.4 Thesis organization	5
2 STATE OF THE ART	7
2.1 Peer-to-Peer (P2P) Networks	7
2.1.1 Concepts and Applications	7
2.1.2 Architecture	10
2.1.3 Effects to the Network Infrastructure	13
2.2 Content Distribution Networks (CDNs)	17
2.2.1 Concepts and applications	17
2.2.2 Architecture	19
2.2.3 Effects to the Network Infrastructure	22
2.3 Client-Server Model	25
2.3.1 Concepts and applications	25
2.3.2 Effects to network infrastructure	28
2.4 Traffic optimization by applications and layer-cooperative approaches	29
2.4.1 Peer-to-peer applications	29
2.4.2 Content Distribution Networks	33
2.4.3 Server-client applications	35
2.4.4 Summary	36
2.5 Application-Layer Traffic Optimization (ALTO) working group	37
2.5.1 Context and Motivation	37

2.5.2	Architecture	40
2.5.3	Viability	44
2.5.3.1	Security	44
2.5.3.2	Privacy	47
2.5.3.3	Incentivisation	48
2.5.3.4	Network Neutrality	49
2.5.3.5	Multi-Domain orchestration	51
2.6	Summary	53
3	SYSTEM ARCHITECTURE AND DEVELOPED MECHANISMS	55
3.1	General Architecture	55
3.2	Role system	58
3.3	Resources	61
3.4	Roles	70
3.4.1	ALTO Client	70
3.4.2	ALTO Server	73
3.4.2.1	Resource Filtering	74
3.4.2.2	Server discovery	77
3.4.2.3	Inter-server communication	77
3.4.3	Network State Provider	79
3.4.3.1	Network Information Aggregator	79
3.4.3.2	Network State Collector	80
3.4.3.3	Network Status processing	82
4	IMPLEMENTATION	84
4.1	Technologies used	84
4.2	Server architecture	87
4.3	Network information aggregator	97
4.4	Network state providers	97
4.5	Server discovery	97
5	EXPERIMENTS	98
5.1	Technologies Used	99
5.2	Setup	100
5.3	Scenarios	105
5.3.1	Scenario 1 - P2P file sharing	105
5.3.1.1	Scenario 2 - HTTP transfer scheduling	106
5.3.2	Scenario 3 - HTTP mirror selection	107

5.4	Results	109
5.4.1	Scenario 1	109
5.4.2	Scenario 2	111
5.4.3	Scenario 3	113
5.5	Discussion	115
6	CONCLUSION	116
6.1	Conclusions	116
6.2	Prospect for future work	116
	Bibliography	117

LIST OF FIGURES

Figure 2.1	Demonstration of Gnutella’s file searching mechanism [13] . . .	11
Figure 2.2	Examples of structured P2P query mechanisms that utilize DHTs	13
Figure 2.3	Examples of <i>Peer-to-Peer</i> (P2P) query mechanisms optimized from their counterparts in Figure 2.2	15
Figure 2.4	Example demonstration of an overlay network and corresponding physical layer [19]	15
Figure 2.5	Conceptual architecture of a <i>Content Distribution Network</i> (CDN) [28]	20
Figure 2.6	Request routing functionality of a CDN [28]	21
Figure 2.7	Client-Server architecture [41]	25
Figure 2.8	Linux Mint prompt to select a software repository mirror . . .	27
Figure 2.9	Approaches to decrease tension between P2P applications and ISPs grouped by their involvement [27]	33
Figure 2.10	ALTO scenario [3]	38
Figure 2.11	ALTO architecture (adapted from [68])	41
Figure 2.12	ALTO services (adapted from [68])	42
Figure 2.13	Conceptual representation of ISP diversity on the Internet . . .	52
Figure 3.1	Conceptual representation of the ALTO system of a given ISP .	57
Figure 3.2	System architecture at a macro level	58
Figure 3.3	High-level communication diagram of a successful resource action request	60
Figure 3.4	Example network topology with ISP boundary	63
Figure 3.5	Example overlay network topology without ISP boundary . . .	64
Figure 3.6	High-level communication diagram of a P2P application utilizing ALTO	71
Figure 3.7	High-level communication diagram of a tracker utilizing by proxy	72
Figure 3.8	High-level communication diagram of a CDN controller utilizing ALTO	73

Figure 3.9	Communication diagram of how external network state providers upload information to the network information aggregator . . .	81
Figure 3.10	Communication diagram of how an ISP administrator pre-processes the gathered network state and uploads it to the ALTO server .	83
Figure 4.1	Controller layer class architecture	88
Figure 4.2	Unversioned service layer class architecture	91
Figure 4.3	Versioned service layer class architecture	92
Figure 5.1	values for different arms (cont.)	104
Figure 5.2	Execution times measured Scenario 1	109
Figure 5.3	Inbound traffic flux by network areas measured in Scenario 1 .	110
Figure 5.4	Execution times measured Scenario 2	111
Figure 5.5	Inbound traffic flux by network areas measured in Scenario 2 .	112
Figure 5.6	Execution times measured Scenario 3	113
Figure 5.7	Inbound traffic flux by network areas measured in Scenario 3 .	114

LIST OF TABLES

Table 2.1	Types of P2P systems (Adapted from [10])	10
Table 3.1	Network node entities in the conceptual ALTO system representation	56
Table 3.2	Example network map referencing Figure 3.4	63
Table 3.3	Example cost map for overlay in Figure 3.5	64
Table 3.4	Example cost map for the limited ISP domain in 3.5	66
Table 3.5	Example endpoint property map for server replicas	67
Table 3.6	Example of an ALTO server's <i>Information Resource Directory (IRD)</i>	69
Table 3.7	ALTO server's available endpoints	74
Table 3.8	Example ALTO queries with the filtering functionality	76
Table 3.9	Network Information Aggregator's available endpoints	80
Table 5.1	Tracker mappings of file fragments to available endpoints	105
Table 5.2	Tracker algorithms to be tested in scenario 1	106
Table 5.3	Measurements to be taken in scenario 1	106
Table 5.4	Dynamically applied client-server path delays	106
Table 5.5	Client algorithms to be tested in scenario 2	107
Table 5.6	Measurements to be taken in scenario 2	107
Table 5.7	Listed server mirrors	107
Table 5.8	Available path throughput from client to mirror servers	108
Table 5.9	Client algorithms to be tested in scenario 3	108
Table 5.10	Measurements to be taken in scenario 3	108

ACRONYMS

ACL Access-Control List.

ADSL Asymmetric digital subscriber line.

ALTO Application-Layer Traffic Optimization.

ANE Abstract Network Element.

API Application Programming Interface.

AS Autonomous System.

BGP Border Gateway Protocol.

CAN Content Addressable Network.

CaTE Content-Aware Traffic Engineering.

CDN Content Distribution Network.

CDNI Content Distribution Network Interconnection.

CORE Common Open Research Emulator.

CPU Central Processing Unit.

DHT Distributed Hash Table.

DiffServ Differentiated services.

DNS Domain Name System.

DoH DNS over HTTPS.

DoS Denial of Service.

DPI Deep Packet Inspection.

DTO Data Transfer Object.

EGP Exterior Gateway Protocol.

EMEA Europe, the Middle East and Africa.

FCC Federal Communications Commission.

GNP Global Network Positioning.

GSLB Global Server Load Balancing.

HTTP Hypertext Transfer Protocol.

HTTPS Hypertext Transfer Protocol Secure.

ID Identifier.

IDMaps Internet Distance Map Service.

IETF Internet Engineering Task Force.

IGP Interior Gateway Protocol.

IP Internet Protocol.

ipv4 Internet Protocol version 4.

ipv6 Internet Protocol version 6.

IRD Information Resource Directory.

ISP Internet Service Provider.

JSON JavaScript Object Notation.

LSPD Label Switched Path Database.

MAC Media Access Control.

MPLS Multiprotocol Label Switching.

MTR Multi-Topology Routing.

MVC Model-View-Controller.

NETCONF Network Configuration Protocol.

NetPaaS Network Platform as a Service.

OSPF Open Shortest Path First.

OSPFv3 Open Shortest Path First Version 3.

P2P Peer-to-Peer.

PaDIS Provider-Aided Distance Information System.

PC Personal Computer.

PID Provider-Defined Identifier.

PoP Points of Presence.

QoE Quality of Experience.

QoS Quality of Service.

RAM Random-Access Memory.

RBAC Role-Based Access Control.

REST Representational state transfer.

RFC Request for Comments.

RTT Round-Trip Time.

SDN Software Defined Networking.

SNMP Simple Network Management Protocol.

SQL Structured Query Language.

TCP Transmission Control Protocol.

TED Traffic Engineering Database.

TLS Transport Layer Security.

URL Uniform Resource Locator.

XMPP Extensible Messaging and Presence Protocol.

1 | INTRODUCTION

1.1 CONTEXT AND MOTIVATION

As society as a whole advances, so does seem to increase the individual's quality of life, which in turn increases the standard to be expected from the society he lives in. As such, technology itself must quickly adapt to the needs of the people it serves, whichever they may be - educational, medical, logistical, just to name a few - and consistently create or improve upon solutions that inevitably change the day-to-day living of the many that use or indirectly reap the benefits of such solutions. A particular example that is still fresh in this generation is in the relationship between people and computers - where they may have been nonexistent a century ago, reserved for industries fifty years ago and valuable household commodity a few decades ago, it is now common to see a family home with more than a dozen computers, with a variety fitting for the many kinds of problems they can solve. The increased number of computers and their expected functionalities has made it so computer networking as a whole has to be improved upon.

The internet allows computers to connect to one another in a worldwide network that applications can use to further increase their possibilities. However, when certain applications go unchecked it becomes very difficult for *Internet Service Providers (ISPs)* because such applications can create traffic which is either impossible, infeasible, or too costly to manage. This issue is further exacerbated when considering the scale of the next decade, where Cisco predicts that by 2022 global internet users will make up 60 percent of the world's population, and global IP traffic will reach 396 exabytes per month [1]. The problem of network management will thus increase in difficulty due to the sheer scale of Internet usage, and traffic engineering solutions are then required to provide certain service standards to applications.

Considering a network of computers which are running applications to fit a given use case for the user, such as transferring a file, watching a real-time video, or consuming the content of a given social network, these applications are responsible for creating traffic that must be supported by the underlying network infrastructure, meaning all the hardware and software that is utilized by given companies to provide to end users

the ability to communicate with each other. These applications can be thought of as citizens of a communications facility that provides the service of accessibility to other citizens, and there is a common incentive in maintaining this facility in such a way that keeps the service up to its standards. As such, and like any other community-shared facility, it must be maintained by the owners, and part of it includes creating and enforcing policies that uphold the facility's quality. During the runtime of an application, the way it is programmed to operate has impact on the traffic it generates on the network, and thus how resourceful it is with the shared domain it uses. The logic of the program dictates how the shared network is used to achieve a given goal, and how it accomplishes it can be more or less preferable by the service providers - for example, application decisions such as which hosts to consume a service from, at what time of the day some traffic is generated, or how much traffic is needed to achieve a use case, have a concrete impact on the shared network structure. *Peer-to-Peer (P2P)* applications are an infamous example of a kind of application that often makes decisions that are not preferable by ISPs. These applications create overlay networks, which are abstract networks constructed on top of the underlying network that supports it, and on which the application's logic runs on, essentially making it infrastructure-agnostic. Historically, P2P traffic has not been preferable by ISPs due to its unpredictable and hard to manage nature. Indeed, if P2P applications simply keep an overlay connection between peers that does not span more than a few hops, whilst ignorant to them being, for example, either direct network links or spanning multiple Autonomous Systems (ASs) in the underlay, the generated traffic is always at risk of being inefficient and too taxing on the supporting infrastructure - for example, by neighboring other peers which reside outside network borders, which are more infrastructurally expensive to reach. As global file-sharing traffic currently uses around 7 exabytes per month (including P2P-based file-sharing) [1], and BitTorrent alone makes up 27% percent of total upstream volume of traffic [2] it's in the best interest of both ISPs and P2P applications a way for the overlay and underlay levels to operate in synergy, i.e., how should the layers combine efforts to guarantee that their needs are met in a sustainable manner.

Current consumer trends suggest that media consumption will make up a considerable part of global Internet traffic. In fact, Cisco predicts that, by 2022, more than 82 percent of all consumer Internet traffic will be dedicated to Internet video streaming and downloads, and *Content Distribution Networks (CDNs)* will carry 72 percent of all Internet traffic [1]. CDNs act by injecting content geographically nearby end users to increase availability and reduce total traffic usage, and are an example of how applications can better leverage the shared domain's resources to achieve their goal.

The CDN's management layer can optimize its application behaviour in ways that are advantageous to both applications using the CDN and the shared network structure, and such ways include what edge server to cache data to, how to efficiently match end users to appropriate edge servers, or how to increase service reachability among other CDNs. Thus, much like P2P networks, content distribution networks could also greatly benefit from cooperative interactions with network providers. These optimizations should be made by the parties which have economical interest in guaranteeing good performance of the overall ecosystem, i.e., those acting on the over and underlay, and should seek to, resorting to application and network administration input, understand how to utilize the given network resources to achieve functional requirements in a way that is cheap, effective and sustainable.

More broadly, most kinds of applications that generate traffic on a network could benefit from input by entities which know how such network is structured and what political and administrative biases exist. Of course, a one-sided approach could also exist to optimize resourcefulness of the network structure - applications could use an independent internal logic that utilizes measurements and its, albeit limited, knowledge of network details to better aid their decisions, and likewise ISPs can attempt to throttle, block, or generally apply traffic engineering. In fact, these one-sided approaches are precisely what happens currently, but this work aims to argue for a two-sided cooperative approach.

In short, the issue that motivates this thesis is the lack of proper cooperation between the overlay and underlay levels in the task of optimizing traffic that originates from decisions that occur at the application level, e.g., peer selection for file retrieval in file-sharing P2P applications, software distribution mirror selection, CDN provider server or cache redirections, high traffic load scheduling, etc. This problem is not new to the *Internet Engineering Task Force (IETF)*, who devised a working group to explore possible IETF standardization on traffic localization after test results concluded that P2P applications that select peers based on exclusive network information provided by ISPs could reduce network infrastructural and administrative costs as well as increase application download rates [3]. Such working group devised a request-response protocol by the same name, ALTO, where clients could query authoritative and trustworthy servers on information that regards to the underlay structure where the client operates. While the tussle between P2P applications and ISPs were the motivation for the ALTO working group to be created, the benefits of a standardized, maintained, and well provided system for network information querying and guidance on traffic-related decisions could help create the vision of ISPs and applications cooperating for

mutual benefit, being thus advantageous for more than P2P applications - in essence, it would be a helpful system for any situation where a decision could be optimized with the addition of proper insight on network infrastructure. This work then focuses on tackling the theme of application-infrastructure cooperation on the Internet, with particular focus on presenting, implementing and improving the ALTO protocol as a cooperation enabler.

1.2 OBJECTIVES

The main objective of this thesis is to develop a working system that adheres to and expands upon the ALTO working group's devised protocol and architecture. The starting point will be a preexisting software project that served as a proof of concept to the strategy of traffic optimization at the application layer, which will now be extended in three ways: firstly, by restructuring and documenting the existing code in order to, through the compliance with the standards of object oriented programming and software development guidelines, present a solution that could be continuously maintained and modified; secondly, by further expanding on the software's functionality, e.g., adding more types of cost metrics, specifying meta-data which give the resources a time-specific applicability, specifying means of synchronizing data among servers, restricting user interaction via access control methods, etc); thirdly, by specifying and implementing a network data supply component to the architecture, as one has not been formally defined by the ALTO working group. Whilst expanding upon the working group's devised solution is indeed a goal, it is also important that the developed work complies with the specifications it is based on, so the work done by the IETF in regards to documentation and general reasoning of the protocol remains consistent with this implementation, with further additions being reasoned in this work.

With the intent of completing its main goal, this work's partial objectives were devised as follows:

- Literature review in regards to application-level traffic optimization and the co-operation - or lack thereof - between overlay networks and the underlay they operate on. More specifically, an understanding of what the problem entails, the consensus on the existing issues, and an overview of currently proposed solutions.

- Complete overview of the ALTO working group's current work. More specifically, an overview of both their existing RFC documents and currently active internet drafts (at the time of writing).
- Familiarization with the existing system to be worked on and definition of both a new system architecture which complies with and extends upon the ALTO solution, as well as the new function modules to be added and how they should operate.
- Implementation of the devised solution.
- Construction of a realistic network simulation scenario and prototype applications to base the experiments in - this includes a P2P file sharing and media streaming application, a server-client file sharing application, and a data center bulk transfer solution.
- Testing of the implemented solution within a simulation scenario, and how it compares with other traditional methods in regards to achieved network infrastructural resourcefulness and client application performance.

1.3 CONTRIBUTIONS

This thesis's contributions include a working implementation of the ALTO protocol as specified by the working group of the same name, which includes functionally extensions, as well as the implementation of a devised architecture to fulfill the ALTO working group's proposed idea of layer cooperation, that includes a network status supply, resource access control, and domain synchronization layers. Additionally, the accomplished experiments in a simulated environment served as empirical proof of the usefulness of an ALTO system for layer cooperation, as it was able to display what is to be gained by using the proposed approach over traditional ones in regards to application performance and optimal network resourcefulness.

1.4 THESIS ORGANIZATION

This dissertation will be organized in six chapters, as follows:

- **Introduction:** Provides context to the tussle between applications and Internet providers, as well as an argument for the necessity to fix this issue to reach a sustainable environment for both parties. Coupled to this, the dissertation's main goal is presented.
- **State of the Art:** Displays the existing theory related to popular technologies or overall concepts that could leverage the ALTO protocol for improved functionality and/or performance; secondly, displays existing proposed solutions to traffic optimization at the application level that do so using network information with and without close underlay cooperation; thirdly, overviews the ALTO working group's proposed protocol and architecture.
- **Specification:** Presents the devised system's functional and non-functional requirements, as well as an overview of the planned architecture.
- **Implementation:** Provides reasoning to the decisions that were made in the task of implementing the specified project.
- **Testing and result analysis:** Overviews the planned simulation scenarios, how they were materialized, how the related tests were performed, and their results.
- **Conclusion:** Presents a critical analysis of the simulation test results, and how they change the previously proposed hypothesis. Finally, it presents the results of this thesis in regards to what objectives were completed, the general conclusions that were retrieved, and future work.

2 | STATE OF THE ART

This chapter aims to provide a literature overview of the topics that relate to the main problem that this thesis aims to help solve, which is the lack of cooperation between applications and the service providers of the infrastructure where such applications reside. As such, the first section focuses on discussing structural network patterns utilized by applications to give them particular properties which are helpful to achieve their use case. Among these patterns, particular interest will be given to three of them for the following reasons: firstly, due to their popularity in the current network paradigm; secondly, due to their potential to optimize traffic that is generated at the application level. Considering this, the patterns that will be discussed are the classical server-client architecture, the distributed approach of P2P architectures, and CDNs, which have been gaining massive popularity in the past years. For each of these, a conceptual analysis is made - more specifically, contextual background, the architecture itself, advantages and disadvantages, and possible use cases. Additionally, there's an examination on how applications that utilize these patterns affect, positively or negatively, the physical infrastructure where they operate on, and where does potential reside for mutually-beneficial cooperative behaviour between these two layers. The following section displays existing proposals for increased layer cooperation, and alongside it a discussion on the practical consequences of adopting them. The final chapter gives special attention to the *Application-Layer Traffic Optimization (ALTO)* working group's proposal for a layer-cooperative system, as it is the baseline for this thesis's work.

2.1 PEER-TO-PEER (P2P) NETWORKS

2.1.1 Concepts and Applications

Due to the many hybrid implementations that have surfaced, the definition of a P2P network has become harder to pinpoint. Nevertheless, a P2P network is grounded on some properties, among them that it consists of many singular computing elements,

the "peers", which have between themselves similar privileges and functions (this contrasts with the client-server architecture, where two different roles exist - the one that provides a service and the one that can consume it - with functionality and control being thus centralized). P2P networks decentralize computational resources as a means to achieve a given task in a way that is inherently different from a centralized counterpart. This decentralized architecture of the entire system as a whole gives it an interesting list of properties, among them:

- **Dynamic scaling:** As all member nodes can share their computing resources with the network, the system increases its capacity with an increase in its users. Since the peers also act as clients to the network, scaling the service becomes less of a challenge as each new client will also act as a server. This then removes the necessity to manage how many service resources are needed - the amount of existing resources is linked to the number of existing clients, and thus there's no need to purchase and manage central resources, as the network dynamically allocates them by nature.
- **Resilience to failure:** Whereas centralized solutions are much more vulnerable to node and link failures, a decentralized one can more easily work around such threats - as all peers can encompass the same server functionality, network services and resources are not provided on a limited set of nodes, but instead can be redundantly deployed throughout as needed.
- **Power decentralization:** As a consequence to sharing server roles and resources, no single peer has direct control of the network, and the information is not centralized. As such, this considerably deters any attempts to overpower the network, e.g., via means of censorship or biased node favouring.

These, however, are not without their nuances - since many P2P hybrids exist, these properties are not immutable. For example, if we consider BitTorrent, which has tracker servers to redirect users to a correct peer with the requested resource, whilst the network itself can still be resilient to failure, the content-retrieval service that the P2P network provides has a single point of failure and of control - the trackers themselves. Furthermore, the P2P network design brings, by its nature, alongside their potentially advantageous properties, also some potentially disadvantageous ones to consider:

- **Security hazards:** The equal functionality property that networks have give peers much power to influence others. Without proper care, malicious peers are a security risk.
- **Management:** Since resources and services are not centralized, tasks such as event logging and resource backups become very difficult, and perhaps impossible if the peers do not abide by any proper orchestration strategies.

P2P applications have had, in the past decades, a mainstream image that is plagued with legality and security issues. Nonetheless, its design possesses many interesting properties - some of them displayed above - that make it fitting for varied use cases, e.g., file sharing, media streaming, social networking, and problem solving with distributed and cooperative algorithms.

Either way, the influence of P2P applications is undeniable: Sandvine [4] published a global Internet phenomena report concluded that BitTorrent [5] alone had in 2019 a global application total traffic share of 2.4%, and perhaps most importantly over 27% of total upstream volume of traffic, and over 44% in *Europe, the Middle East and Africa (EMEA)* alone [2]. Beyond file sharing purposes alone, P2P applications have been recently considered a fitting solution for low-cost content delivery systems in high demand scenarios - for example, in applications such as PPStream [6] in China, which provide television content over IP to large audiences. Similarly, Akamai [7] recognizes the potential of P2P technologies to provide a highly distributed option for serving static content over the network, although it being currently lacking in management and control features [8]. Indeed, P2P Internet video broadcast services - and world-wide static content delivery services for that matter - seem attractive as they are cost-effective and easy to deploy, and are fitting to large scale demands, and thus have the potential to become a more mainstream solution [9].

Concluding, the P2P network architecture has many fitting use cases, and their rather different strategy, compared to the client-server architecture, to achieve its goals gives it many potentially interesting properties for users and ISPs. Considering its large global traffic share, particularly in upstream traffic, and its potential adoption towards the large scale demands of the future, P2P applications are likely to persist and will be in the minds of ISP administrators for the near future.

2.1.2 Architecture

As stated previously, the term "Peer-to-peer" has become very broad and now serves as an umbrella for many different variations of the core decentralized architecture design. Thus, this chapter focuses not on giving an overview of a single conceptual architecture of what defines a P2P network, but instead of the many existing variations and how they differ among themselves. All P2P networks are characterized by consisting of peers that know one another as to form a so-called overlay network on top of its supporting underlay network. How peers are organized in these P2P networks and how they operate is what distinguishes the many sub-types. Table 2.1 groups known P2P systems in regards to their centralization and structure, similarly to the groupings of [10] and [11], with the latter further distinguishing the protocols in regards to other parameters, e.g., security, reliability, and performance. The rest of this section follows the survey made by the former.

One would expect that all P2P applications would have no centralization at all, since the P2P design ponders functionality spread throughout the network. However, some modifications have been made in some of these sub-types, which shift how much decentralization they really have in practice. Similarly, different strategies exist that dictate the structural hierarchy that resides within the member peers. As would be expected, these sub-types of P2P networks thus possess different strengths and weaknesses, and these can be leveraged to the most appropriate use cases.

Table 2.1: Types of P2P systems (Adapted from [10])

		Centralization		
		Hybrid	Partial	None
Structure	None	Bittorrent, Napster, Publius	Kazaa, Morpheus, Gnutella (extension proposals), Edutella	Gnutella, FreeHaven
	In Infrastructure			Chord, CAN, Tapestry, Pastry
	In System			Bittorrent (DHT/Trackerless), OceanStore, Mnemosyne, Scan, PAST, Kademlia, Tarzan

Early versions of Gnutella [12] come as a famous example of a decentralized and unstructured architecture, as peers act with equal functions and privileges, and no inherent structure exists for peers to connect to each other, nor does it for storing or retrieving content on the overlay network. The bootstrapping phase consists of users reading from list containing a set of Gnutella peers - a list that is retrieved from a trustworthy source - and attempting to connect to each one of them until a preferred number of known neighbours is reached. The unstructured nature of this protocol makes it so there's no systematic way to efficiently retrieve content, and thus a technique of flooding the network with content queries is used until either a reply is met or the predefined TTL value is exceeded, as is exemplified in Figure 2.1.

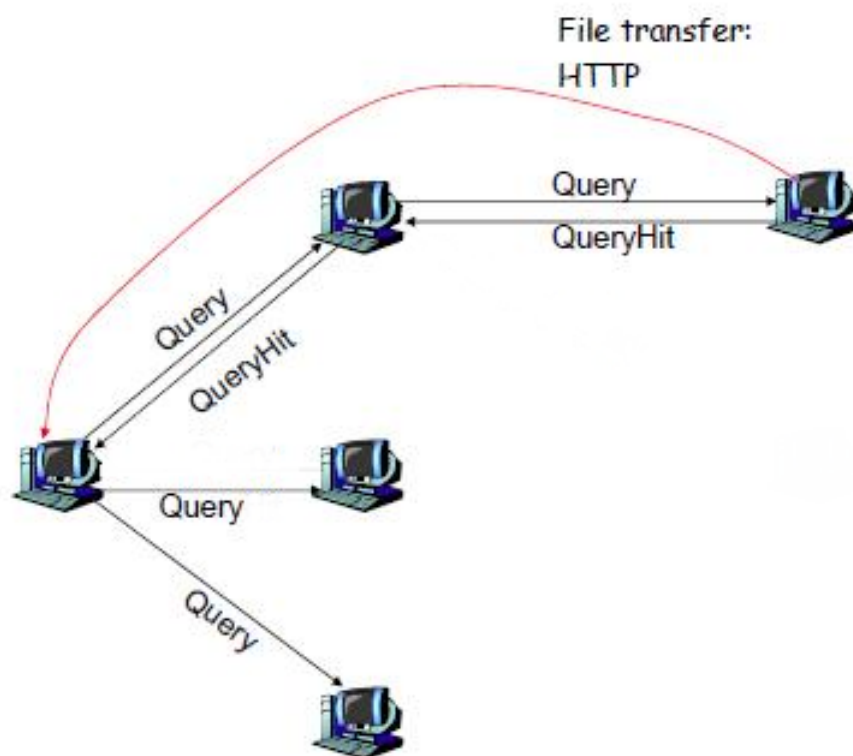


Figure 2.1: Demonstration of Gnutella's file searching mechanism [13]

Partially centralized architectures were defined as similar to those which are decentralized, but with the added caveat that some peers are chosen to service a portion of the network. This is done to take use from the fact that not all network peers are alike in terms of memory, computational power, or other relevant resources. As such, more capable peers are elected as "supernodes" and are delegated with more responsibilities,

noting that the network self-configures in situations where such elected nodes either fail or willingly leave the network, and as such there is no single point of failure as there would be on a true centralized architecture.

A hybrid architecture approach in a P2P network employs some elements from the client-server architecture. With Napster [14] as an example, whilst peers still operate as servers or clients, they must contact an intermediary and central server when querying for content, which will in turn redirect them to one or many peers that contain it - a similar concept applies for BitTorrent, where such intermediary servers are called trackers. Obviously, the choice to add a centralized aspect to the architecture hinders many of the advantages from a purely unstructured solution - namely its scalability, resilience to failure, and decentralization of control - serving as a trade-off to facilitate the control and maintenance of the network, as well as the peers' ability to bootstrap to it and locate content.

A P2P architecture is structured if it employs some non-random and systematic criteria on how the network operates, e.g., how peers organize themselves and where content is stored and how it is retrieved. For example, FreeNet [15] uses the content's hash as a key that is used to query for it, and which in turn is used by the peers in each subsequent hop to know where to forward the request, instead of flooding the network in attempts to blindly find it like Gnutella does. Many of the structured P2P architectures rely on *Distributed Hash Tables (DHTs)*, which act as a decentralized map structure that binds a given key to some content in the network, in such a way that the full key-space is partitioned over the peers. Two examples of structured P2P architectures that use DHTs can be seen in Figure 2.2 - to the left, the Chord algorithm uses a circular DHT where each peer knows the location of some peers that are their predecessors, and some that are their successors. When a peer needs to query for some content, it uses its key to firstly search for it locally and, if it doesn't exist, it forwards the query to their successors, and the process recursively continues. To the right, the *Content Addressable Network (CAN)* has the key-space mapped to a virtual two dimensional grid, and its area is partitioned to peers considering a deterministic function, which in turn is used by querying peers to figure out where a given content is stored. A straight arrow from querying node to providing node represents the routing path that the querying message must travel: A-B-E.

Employing a systematic way to self-organize and share content is the means to guarantee that a P2P network can be fully decentralized whilst maintaining a desirable level of performance. However, the reliance on structure means that it must be maintained, e.g., managing neighbour pointers on Chord or managing area allocations on

CAN, and that can be costly or even impossible with high rates of peer churn, i.e., with a sufficiently large rate of peers entering and leaving the network.

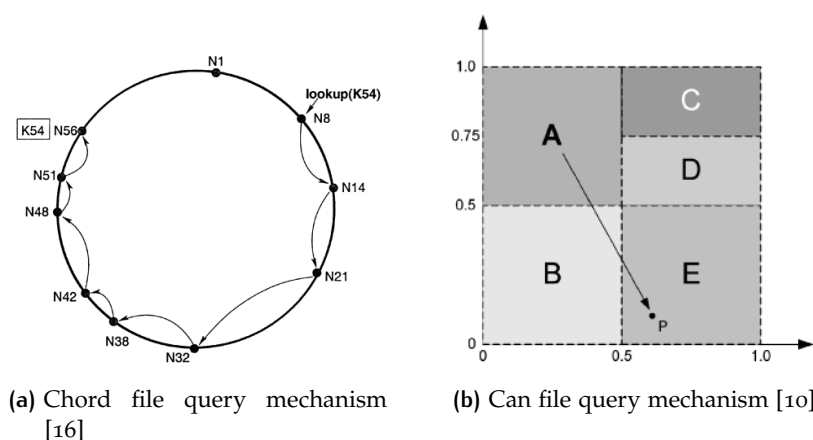


Figure 2.2: Examples of structured P2P query mechanisms that utilize DHTs

2.1.3 Effects to the Network Infrastructure

Historically, ISPs have deemed P2P traffic as unideal or even undesirable. Besides the aforementioned illegality precedent that is tied to P2P applications, the overall properties of P2P networks make them unappealing to support - due to the distributed nature of these types of networks, the overall traffic is less predictable, with the higher upload traffic volume in edge networks requiring infrastructural investments, and the network-agnostic operation mode of P2P applications leads to inefficient and uncooperative network resource usage.

P2P networks who neither have structure nor a central point of control have to utilize content retrieval methods which are bound to be less efficient than their counterparts. However, architectures which fit in these categories mostly do so with a clear purpose - Gnutella's decentralized nature makes it very hard for individual nodes or external entities to regulate what can happen in the network (such as enforce legal actions), and its lack of structure simplifies the implementation and reduces the overall effort to bootstrap to the overlay, making it a good fit for applications with a high peer churn rate. Similarly, FreeHaven, an also unstructured and decentralized P2P protocol, has its architectural decisions fit a very specific use case, as it "emphasizes distributed, reliable, and anonymous storage over efficient retrieval" [17]. The lack of systematic means to efficiently locate content by these P2P architectures means that more ad-hoc

methods have to be used, which are less efficient and thus incur in bigger workloads for ISPs - the usage of query flooding by Gnutella and message broadcasting by FreeHaven are examples of this.

The usage of structure by P2P networks can, as stated before, result in more efficient content and peer location algorithms. However, maintaining such structure also requires a chunk of ISP resources, as peers need to periodically update other neighbouring peers, as well as react them abruptly entering and leaving the overlay. The usage of key-value mappings with DHTs can also have the potential to be ISP unfriendly, as the hash function can be designed to evenly distribute resources over the peer pool, and thus over the network - whilst such property is certainly advantageous in certain use cases, doing so removes any application context that exists in the content - for example, grouping resources which belong to the same web page with such a method isn't efficient, as they will be individually hashed and spread throughout the network, despite the fact that they'll likely be requested together for each page access.

A first point of improvement is optimizations made in the applications themselves to less degrade network resources. An example of these can be visualized in Figure 2.3. To the left, a point of optimization in the Chord system would try to reduce the number of query messages per resource by increasing the number of successors a given node knows. That way, the querying node can instead query not for the single successor it knows, but instead by querying for the one who's ID immediately precedes the content's, thus insuring a reduced number of hops to retrieve the message. This consequentially also reduces the total amount of traffic on the network, and improves application times. To the right, a point of optimization in the Gnutella system could try to tackle the usage of query flooding to locate data, as such flooding grows exponentially and thus intakes a massive toll on network resources. A query flooding system would not be as prejudicial if content was equally scattered throughout the overlay and each given content was a minimal amount of hops away. However, as concluded by extensive analysis of user traffic on Gnutella during its heightened use, nearly 70% of users shared no files and nearly 50% of all responses were returned by the top 1% of sharing hosts [18].

Regardless of the many ways through which glsp2p systems can operate, e.g., in regards to structural mechanisms and centralization, and even disregarding potential application optimizations, no classic P2P system operates in full understanding of the underlying network topology, nor with a cooperative behavior towards ISPs. The network-agnostic manner under which they operate results in overlay networks which are layered on top of the underlay where they run, as exemplified in Figure 2.4 - as P2P

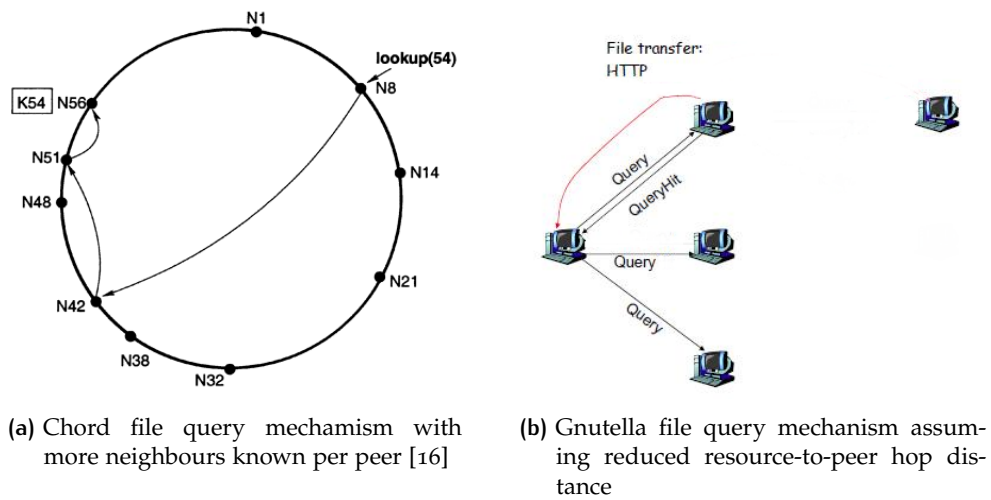


Figure 2.3: Examples of P2P query mechanisms optimized from their counterparts in Figure 2.2

applications are network-agnostic, two neighboring peers could exist in completely different contexts on the common network layer - for example, they could either be connected by a single data link or be multiple network provider domains away from each other.

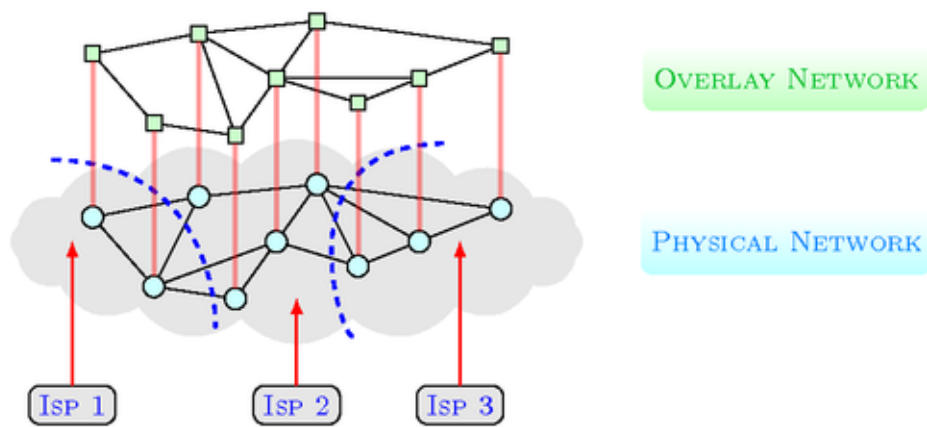


Figure 2.4: Example demonstration of an overlay network and corresponding physical layer [19]

P2P application's inability to localize traffic is seen as a big problem - as concluded by [20], ISPs face bottleneck bandwidth pressure in the large scale Internet of the future, with it being in particular from P2P applications. However, an argument is made

that an increase in users from such applications is not necessarily harmful, and perhaps helpful, if traffic locality can be boosted. Thus, there is a necessity to ensuring that peers prefer to generate traffic within network locality over traffic that crosses network borders. A first step would be increasing the availability of content inside network boundaries, through means such as cache injections or peer content serving incentives. Given that content resides locally, a second step would be that P2P applications have a network-aware vision of the overlay, which does not happen with classic P2P protocols. This issue is particularly damaging in P2P applications whenever neighbours are selected, and whenever a service exists redundantly in the network and a decision must be made in regards to what peer or collection of peers to consume the service from. If it indeed is the case that P2P applications are not locality aware, it can easily be seen how this can be an issue: for the applications, choosing peers which are not local to the querying peer may result in more time to retrieve the requested contents; for the ISPs, bad network resource utilization can incur in higher operational costs, and may degrade overall network performance in many cases, e.g., by not being able to stop applications from overusing inter-AS links, which usually are network bottlenecks (a conventional wisdom demonstrated in [21]) and which, due to peering agreements with other ISPs, are less desirable to be overused due to peering costs. If the P2P application were to attempt to choose the peers that would most effectively serve the querying peer, it could depend on privileged information that the ISP has on the network's properties and current status, such as the inherent network topology, link properties, or scheduled server maintenance. Attempting to optimize peer selection without a co-operational channel with ISPs would be sub-optimal, as not enough information can be derived with proving alone, and could perhaps even be more damaging with the wrong techniques - consider a peer selection algorithm that chooses the peers with lowest *Round-Trip Time (RTT)* of a probing ping message, whilst having no indication on available end-to-end bandwidth, existing network bottlenecks, or peak traffic hours. Likewise, attributing neighbour pairings via geographical proximity, whilst initially seems like a good step in location awareness, may also not be optimal - ISPs may not always prefer geographical proximity in connections, as peers could be very geographically close but residing in different *Autonomous Systems (ASs)* and thus separated by costly links. Other peer-selection techniques focus on randomly selecting nodes, such as the means through which a BitTorrent tracker selects between redundant peers serving the same file chunk [22], which is simple and resilient to peer churn [23], but as a consequence is sub-optimal on network resource usage as no network consideration exists. It is reasonable to assert that no P2P application can act

with full ISP consideration without directly cooperating with it, and simple heuristics should be, whenever possible, traded for methods where full cooperation with the underlay is done - that is, if the needs of both layers are being met.

Indeed, it is the case that current P2P solutions are ISP-unfriendly. More concretely, [24] shares the view that P2P applications and ISPs are in a tussle, since P2P applications generate traffic which favours the application's needs whilst ignoring the ISP's, which in turn upsets the latter's business model. To name a few examples, BitTorrent seems to employ peer selection algorithms which do not consider the underlay network, which can result in degraded download performance and increased load on ISPs [23]. [25] found that since this protocol is locality unaware, 70-90% of existing local content was found to be downloaded from external peers, and suggests that locality-aware content distribution algorithms could significantly reduce the total amount of traffic generated. Likewise, Gnutella generates traffic which is not ideal, as it may have to cross ISP network boundaries multiple times [26] due to the same fundamental issue stated before - an application layer that operates in disregard to the network underlay it runs on.

As [27] describes, the ongoing friction between the overlay and underlay layers has made it to the point where ISPs have chosen to throttle the bandwidth of P2P traffic, or even outright block it. In return, P2P applications have tried to mask their presence to bypass such restrictions via tunnelling or using non-standard and random port numbers. This is an unsustainable system that is bound to hurt both ISP profit and application functionality, and a strategy of cooperation between the overlay and underlay layers is crucial to guarantee that the requirements of both parties are met, specially in the face of the increased technical and scale-related challenges of the future.

2.2 CONTENT DISTRIBUTION NETWORKS (CDNS)

2.2.1 Concepts and applications

A CDN, as the name implies, is a network specifically designed with its main focus on distributing content to a set of end users. Its design allows for the alleviation of performance bottlenecks on the Internet generated by client requests, and has been recently been considered a powerful tool as a response to the existing high demand for media content, which has a huge share of the global Internet traffic of today.

Functionalities of CDNs include [28]:

- **Content outsourcing and distribution:** Replicate content throughout the network into edge servers, which are deployed nearby end users. This allows CDN clients to pay for their content to be hosted on these edge servers, and in doing so guaranteeing that it is quickly accessible by end users.
- **Request redirection:** Direct a content request to the most appropriate edge server at a given time. This redirection is done considering relevant network properties, such as geographical locations and current server loads.
- **Content negotiation** Manage the network's properties to fit the needs of its clients through negotiation.
- **Management:** Manage the distribution network, which includes accounting, monitoring, statistical analysis on content consumption, etc. A close management of the distribution network is important for its business model, as, besides being needed for a billing system, allows for a better understanding of the networks' usage patterns, which is helpful information for better engineering the system to most optimally serve content with increased revenue and decreased costs.

The current focus of CDNs is thus to provide content, e.g., web pages, documents, photos, videos, or media-related streams, with high availability and performance. The strategy used by them to guarantee a satisfying *Quality of Experience (QoE)* on a global scale is the deployment of content close to the end-user - a CDN contains many nodes which are geographically spread throughout the globe and close to the users they wish to serve, and whenever such users request for content, they are routed to the node which is closest to them ([29]). Data replication to servers which are strategically placed closest to end-users, coupled with good means to properly redirect such users to the most attractive edge server, is what allows content to be available more often and more quickly. These are undoubtedly attractive features in the world of e-commerce, where user QoE can dictate much of the profit - for example, Akamai, one of the leaders in CDN-related services, ran a research concluding, among other things ([7]):

- A 100 millisecond slower web page loading speed can result with a 7% drop in sales
- A 2 seconds slower web page loading speed can almost double the number of visitors who end up abandoning their carts

- 53% of users who use smartphones to visit web stores won't make the sale if the web page takes more than 3 seconds to fully load
- 28% of users won't return to the same web store if they think it takes too long to load
- A 250 millisecond faster loading time proved to keep users from visiting a competitor web store

It should then come to no surprise that streaming services such as Netflix [30] and Youtube [31], who now reach a global scale and whose utility is highly dependant on their high availability and low transmission delay, routinely use CDN solutions. More broadly, typical CDN customers include media and Internet advertisement companies, data centers, ISPs, online music retailers, mobile operators, etc [28]. Indeed, companies that wish to provide a given service in the web at scale routinely partner with companies whose focus is providing content delivery services, with popular examples being Akamai, CloudFlare [32], or Amazon Cloudfront [33]. Coupled with the promise of highly available and quick content retrieval, these companies also couple other attractive services, such as firewalls and *Denial of Service (DoS)* protection. The Internet's currently most targeted use being media consumption has made it so CDNs and their providers have an important role in dictating a very considerable percentage of flow of traffic in ISP-owned infrastructures, and as such their study and improvement is quite important, as are the efforts to increase harmonious behaviours between content delivery applications and service providers, with the goal in mind being network resource efficiency to guarantee that ISPs can remain operational and applications can provide a satisfiable user experience.

2.2.2 Architecture

Figure 2.5 represents a high-level conceptual architecture of a CDN. The true power of a CDN comes from its strategically deployed cluster of replicated servers - at a global scale, this implies having them geographically dispersed and located inside or nearby networks with large content demand. The origin server possesses the content that is to be served, and the bootstrapping process has it uploaded into the network, which is afterwards replicated into these edge servers.

Figure 2.6 displays how, conceptually, the request routing functionality of cdn. As can be seen, the request is firstly directed to the origin server, and serves only light,

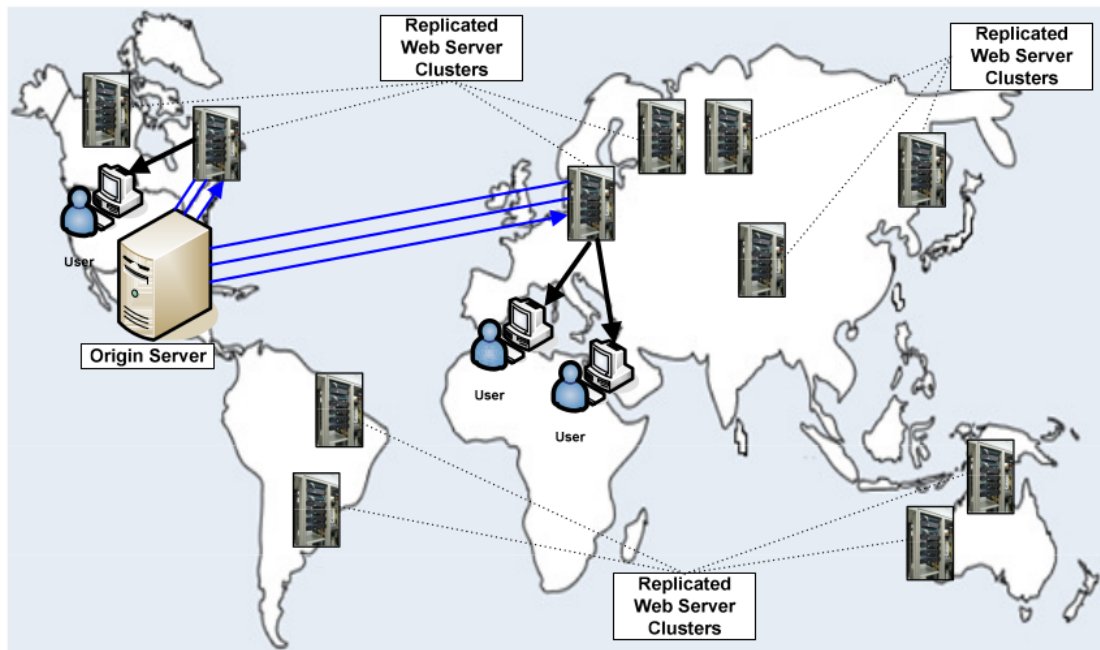


Figure 2.5: Conceptual architecture of a CDN [28]

basic content. In the situation where large static content is requested, the origin server redirects the request to the CDN provider, which utilizes a selection algorithm to elect the most appropriate edge server to serve the content to the end user.

The request routing mechanism is the one that redirects a given user to a given edge server. The prevalent approaches are, according to [34], the following:

- **Domain Name System (DNS) request routing:** The user must first resolve a domain name to retrieve a content. The CDN's DNS processes the request and, utilizing the user's *Internet Protocol (IP)* address, historical measurement information and current server loads, responds with the address of the edge server that seems most fitting to provide such content.
- **Hypertext Transfer Protocol (HTTP) request routing:** Content is firstly requested to a nearby proxy server, which in turn answers with an HTTP redirect to be resolved by the client in order to find the content. The HTTP requests can occur in subsequent rounds and can also use DNS knowledge when the redirection domain must be resolved.
- **Anycast request routing:** The CDN provider announces an anycast prefix to the network. Whenever a router receives multiple announcements to the same

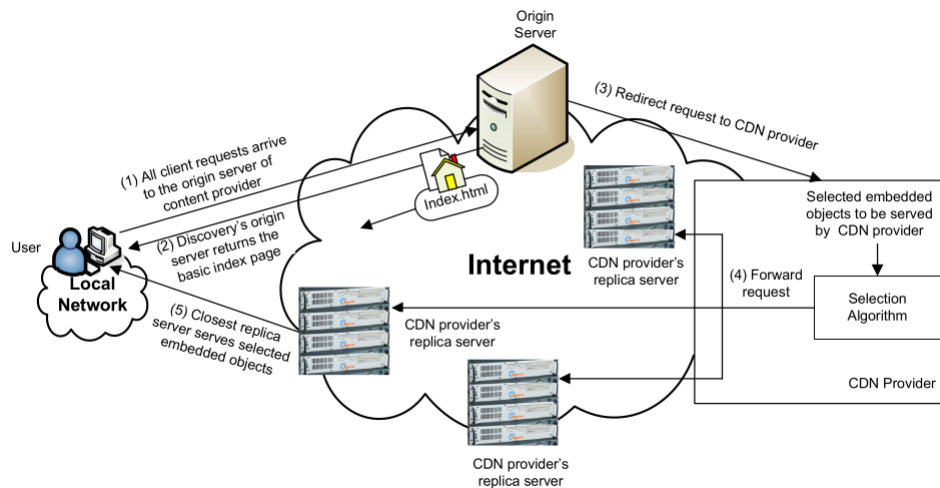


Figure 2.6: Request routing functionality of a CDN [28]

prefix incoming from different locations, it chooses one considering some custom criteria, usually being AS hop count and *Interior Gateway Protocol (IGP)* weight. Thus, different routers can have a given anycast address mapped to different hosts, meaning the ones that are most suitable for that particular router.

These mechanisms are also discussed in [28], but also add:

- **Global Server Load Balancing (GSLB):** Service nodes, consisting of edge servers and GSLB-enabled web switches, are interconnected in a network. Individual nodes possess global awareness of the network, meaning the status of each individual server. With edge servers having more information on the health and status of all other candidate edge servers, and the web switches acting as authoritative DNS servers, the network can enable the support of a global-wide server load balancing mechanism that intakes dynamic server information.
- **Uniform Resource Locator (URL) rewriting:** The origin server redirects the end user via dynamically altering the host pages' URL links.
- **CDN peering:** An extension of a single CDN network to multiple, interconnected CDNs which serve content on behalf of others when appropriate. This is helpful to extend the domain reachability of a single CDN, increase fault tolerance, and ability to achieve better performance with more candidate servers to choose and balance loads from.

It is vital that a CDN possesses a clear view of the network performance inside its domain. [28] lists important metrics which are used to measure CDN performance:

- Geographical proximity of end users
- Path latency
- Path packet loss
- Path Bandwidth
- Path startup time
- Path frame rate
- Server load

Means through which these metrics are retrieved by the network include traffic monitoring of end user to surrogate server communications, and surrogate server feedback retrieval via application requests or measurement probings [28] [8].

Having a clear understanding of CDN performance is important for system management in two fronts - firstly, through performance evaluation, by providing the billing and monitoring modules the verification of how the network is faring at its task of caching and delivering content on behalf of clients; secondly, through performance optimization, by providing the logical layer an updated view on network status, serving as contextual input that the system uses to better reason on how to act - for example, in regards to the caching and request routing mechanisms.

2.2.3 Effects to the Network Infrastructure

As previously discussed, CDNs came as a tool to strategically position content on the network in such a way that it can more quickly and more reliably be retrieved by an end-user. These CDN systems are then inherently a means of optimizing network resource utilization in the application layer, and thus are of great interest for ISPs and, if done properly, can be very appealing not only to them but also to end-users that use applications leveraging these networks.

The usage of a single content-providing server - or a limited set of them - which is far away from the content demand that in turn has a large geographical distribution is prone to server overloading and path congestion problems if a big enough scale is

achieved. The usage of data caches is a classic solution for network inefficiency problems, which is used by CDNs as a means to replicate content to strategic locations to better serve users, with the added benefit for ISPs that their network resources are efficiently used, with the ability of reducing the total amount of used bandwidth needed for a service to operate - as data travels a shortened total amount of network hops from data source to points of data demand - and reducing congestion of inter-domain links - as data caches will reside locally and redistribute traffic away from highly shared network links. It can thus be stated that the relationship between CDNs and ISPs allows for a win-win scenario because efficient network usage has consequentially better service quality, benefiting service providers and applications, respectively.

However, CDNs seem to lack cohesion with the underlying network infrastructure during normal application operations. As stated by Akamai, a leader in CDN services, in their report [8], the large scale and complexity of the Internet, where it takes well over 650 networks to reach 90% of all access traffic, adds to it many challenges to the CDN's role of content delivery - in particular, whilst good investment seems to exist at the first and last mile of internet access (website hosting and end users, respectively), there seems to be little economic incentive to invest in the middle mile, composed of peering points shared among networks, which in turn become network bottlenecks and become susceptible to increased traffic packet loss and latency, making inter-network communications unreliable, and loose coordination between autonomous networks with internal biases is pointed as a main cause. Due to this, even a well provisioned data center will be at the mercy of the various inter-network bottlenecks that may arrive, and performance takes a hit. In fact, the paper suggests a clean-slate redesign of the Internet as a potential solution to its many problems - besides the peering point congestion mentioned above, inefficient routing protocols, unreliable networks, inefficient communications protocols, and application limitations also add to the problematic - but such a redesign to a massive and highly invested global infrastructure doesn't seem plausible.

In alternative to proper network infrastructural insight, CDNs have to rely on network probing, traffic monitoring, and server feedback, as discussed on Section 2.2.2. Even assuming that these are sufficient, the usage of probing techniques will incur in overhead traffic on the network, with this overhead being exacerbated if many other overlay networks or applications also apply this technique. Similarly, traffic monitoring to extract end-to-end path metrics to end users requires resources and takes time, and may also incur in redundant operations being applied on the network. Such

advantage goes outside of the CDN realm, being useful for any overlay-residing application that wants to utilize network probing to be more underlay-aware.

Advantageous as network probing and traffic monitoring mechanisms can be for CDNs to properly conduct request routing and caching decisions, a case must be made for proper application-infrastructure synergy during decision making in the overlay. Much like P2P systems, to infer on network status by measuring it - versus receiving input by trustworthy, authoritative, entities that possess privileged network information, such as ISP administrators - is insufficient. Attributing node pairings - in CDN's case between edge servers to end-users - entirely on geographical data was previously discussed as being a non-optimal way of assessing node selection at the application layer in Section 2.1.3. Again, much like in the scenario of peer selection in P2P systems, the usage of network measurements made by the CDN itself to better pick the appropriate end-server, while potentially be beneficial, can certainly be improved upon if it used additional, hard to retrieve data that only ISPs or other privileged infrastructural entities could possess, and which are at position to guide applications in the infrastructure from whom they have detailed knowledge.

There indeed seems to be a consequential coupling between overlay decisions in the CDN systems and the underlying infrastructure - were the CDNs not to take ISP input when redirecting clients, suboptimal choices would be made that would be prone for bottleneck congestion, and would ISPs employ only their own biases in content request redirection, user-level application *Quality of Service (QoS)* agreements might not be met. [35] states that this lack of awareness to network status is indeed problematic for CDN systems, listing end-user mismatching to edge servers based on dubious DNS-based location binning and resource consuming, non exact methods to detect bottlenecks, as well as lack of agility in server deployment in ideal locations. This is a view shared by [36], which adds that these problems reside in a shared medium that raises the opportunity for cooperative behavior that would enable better application performance and optimized ISP resource utilization. In fact, Akamai themselves have formed content delivery strategic alliances with major ISPs, with AT&T [37], Orange [38], Swisscom [39] and KT [40] among them [35], which hints at this type of partnership being the norm for content distribution technologies.

ISP input permits applications to act in a more network-aware fashion than without it - whereas pairing overlay nodes or deploying edge servers in terms of geographic distance, RTT distance, or any other metric, may give further decision power than a purely network-agnostic overlay system, proper guidance by ISPs could help pairing based on more complex metrics that, besides the aforementioned ones, also consider

ISP objectives - for example, to minimize network distance, avoid bottlenecks, locate content caches, etc. This is further heightened if many other overlays coordinate their efforts with the ISP, which can now in turn orchestrate its traffic, which would previously be generated with no input from it, in such a way that maximizes network resource utilization and, consequently, application performance - this is an important point to consider because one-sided application decisions cannot fully grasp network status nor can they coordinate efforts with other applications for a more harmonious, and thus efficient, Internet.

2.3 CLIENT-SERVER MODEL

2.3.1 Concepts and applications

The Client-Server model is a classical way of attributing roles in the network. Whereas the P2P architecture blends server and client roles into each node, this architecture delineates two distinct ones - the client and server - and their purpose on the network, as Figure 2.7 shows. With it, the service layer is centralized onto the server nodes and the only role expected of client nodes is to request services from them. Operating with opposite philosophies from the P2P architecture, it is to be expected that the advantages and disadvantages should too be contrasting - by centralizing the services, maintenance and general administration of serving nodes become attainable, and by limiting the number of servers to a select - and trustworthy - few, instead of scattering that functionality to all nodes in the network, greatly minimizes security risks. On the other hand, the drawbacks are also apparent and mirror a distributed solution - issues of scalability, resilience to failure, and resilience to monopolization of power arise from the decision to engineer an application that creates clear boundaries between client and server.



Figure 2.7: Client-Server architecture [41]

This classic architecture has seen a lot of adoption by applications, with use cases that include serving content via HTTP or , or enabling e-mail communications [41], to name a few. Adding to this, the client-server approach also serves as a direct alternative to the P2P one in many fields, e.g., file transfer or media streaming, with their respective advantages and disadvantages needing to be weighted out for a proper choice to be made.

As an attempt by service providers to still operate a client-server model whilst reducing its associated downsides, given techniques were created and employed. One of which, by the name of CDN, is of particular importance in this work and as such has its specific overview in Section 2.4.2. Some others will be discussed below.

Server mirroring, one famous technique, is the act of continuously synchronizing a server into its replica, or mirror, essentially creating an exact copy of it that is now accessible as if it were the original. Whilst CDNs aim at replicating chunks of contents wherever it may be necessary, the act of server mirroring performs an integral copy of a server which is self sufficient at servicing a given client, as long as it periodically checks up with the primary server for synchronization. It is a standard business strategy that uses redundancy as a means to increase reliability, availability and performance. The existence of many servers that perform the same task means that these can be strategically chosen to serve a client in a given situation, e.g., by selecting the one that has reduced load and smaller end-to-end message delay to the user. Figure 2.8 shows an application of this, where multiple server mirrors exist to deliver software packages to the Linux Mint [42] distribution. The user has the choice to manually select one of these mirrors, and ideally chooses the one that is most fitting to them.

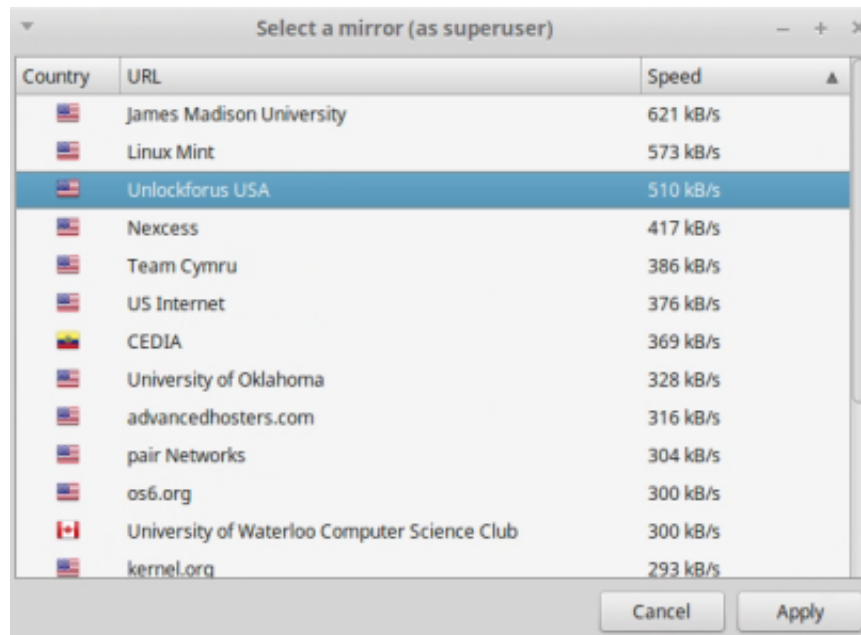


Figure 2.8: Linux Mint prompt to select a software repository mirror

Server load balancing is another popular technique to reduce server-client down-sides, and it refers to the act of distributing tasks over a set of computing nodes as a means to make the service more efficient, since a server with little processing load is bound to respond quicker than one with more, and situations of overloading become less common as it is now spread out over multiple processing units. Load balancing algorithms can be categorized, according to [43], in two categories:

- **Static:** Does not take current system status into account, instead acting upon static rules that define the load type, in regards to, for example, processing power and memory requirements. A round-robin or random approach of load attribution are part of a static approach.
- **Dynamic:** Load is distributed at run time, considering dynamic information about the system that is continuously being collected.

A static approach can be compelling due to its simple nature that makes it easy to implement and much faster in comparison to run. On the other hand, a dynamic approach is more complex by definition as it requires the additional infrastructure required to collect and compute upon the system statistics. As it might be expected though, a load balancing scheme that attempts to optimize resource utilization considering real-time information about system status and performance is bound to have

bigger potential at effectively selecting which server should handle a given request since more factors are known to better aid the decision, including the current server loads in the server cluster, their available resources, network conditions, etc.

Several methods exist through which to implement load balancing. For example, DNS-based approaches perform the load balancing at the name resolving stage, where the IP address response is given after selecting an optimal server, such as the proposal in [44]. *Software Defined Networking (SDN)* solutions, as another example, leverage a control pane that is responsible for intercepting server access calls and activate the load balancing mechanisms that redirect to the selected server, as happens in the proposal in [45].

2.3.2 Effects to network infrastructure

As a base method, the utilization of the client-server architecture is not at all recent to ISPs. Quite the contrary, it is the norm throughout the decades to provide a service by using a set of single-purpose server machines. Intuitively, using a limited number of servers to respond to clients needs will have problem scaling, as increased demand will overload servers and create path congestion, hence the need to employ some of the strategies discussed above. These strategies come from the necessity to fulfill both application and ISP requirements - i.e., to achieve proper QoS levels and infrastructure resourcefulness that helps the general good function of the network, respectively.

Much like the content replication utilized in CDNs, an integral replication of a main server proves itself as an advantageous tool capable of delivering services more closely to users, and as such allows the reduction of total amount of bandwidth used to serve all clients. Optimizing application traffic is crucial to guarantee good network resource usage, and in case of server mirroring it comes down to good strategical deployment and dynamic, intelligent algorithms to properly attribute mirrors to requesting end-users. Giving end-users the choice to manually select the serving mirror, as commonly happens when downloading software, for example, seems problematic, as application-generated traffic is not optimized. In fact, when considering the Linux Mint software package distribution discussed above, despite there currently existing seventy mirrors deployed throughout the globe to fit this role, a large number of these remains mostly unused whilst the main and default server is constantly prone to overworking [46]. It can be stated that end-users both don't care enough to optimize traffic nor do they have enough information to properly do so even if they did. Deployment of server mirrors

is a great tool that brings with it the issue of optimizing server selection, and much like all examples given so far, traffic generated by applications can be firstly optimized by the applications themselves if they consider static and dynamic information of the network they operate on.

Load balancing is too a point of great interest in the task of application-layer traffic optimization. The task of attributing a current request to a pool of available redundancy workers is a common strategy to help scalability, and how it is done has concrete network consequences as it has the power to shape great amounts of traffic in ways that can be more or less preferable for ISPs. It could then be deduced that a load balancing strategy has better chances of being efficient if it fully aware of both server status and network conditions. Doing so with ISP cooperation would reveal insight in some network status information and preferences that could not be retrieved otherwise, and, much like what was argued for in section 2.2.3, could severely reduce overhead traffic created by probing and measurement strategies if such data was centralized in an entity with full network knowledge.

2.4 TRAFFIC OPTIMIZATION BY APPLICATIONS AND LAYER-COOPERATIVE APPROACHES

This section serves to display proposed solutions and existing implementations that have been made in the attempt to optimize application traffic utilizing network information. Given the increasing scale of the Internet as a near ubiquitous system, and the increasing tension between applications and service providers, it comes with no surprise that the area of layer cooperation has been through exhaustive work. Many solutions have been devised for specific use cases, with varying degrees of power given to each one of the layers, and different levels of cooperation. Along with their description, their review is made in regards to what the proposals state their impact is to applications and the infrastructure, as well as the accompanied advantages and disadvantages.

2.4.1 Peer-to-peer applications

Many different mechanisms have been developed with the goal of decreasing tensions between ISPs and P2P applications, which is a subset of the general layer co-

operation problem. Figure 2.9 represents a grouping proposed by [27] where such mechanisms are ordered in agreement with how much involvement the P2P systems and the ISPs have. These classes are as follows:

- **Class 1:** There is not much - or any - interference in the overlay by ISPs nor are P2P systems cooperative. Instead, ISPs apply traffic engineering methods to selectively favour types of traffic. This is usually done to guarantee certain QoS levels to some classes of traffic, which are then to be treated favourably at the forwarding and routing levels. Examples of such techniques are *Differentiated services (DiffServ)*, *Multi-Topology Routing (MTR)* and *Multiprotocol Label Switching (MPLS)*. These classes of methods do not fix the underlying application behaviour, but are instead used to control preexisting traffic. As such, the peers' routing decisions are not affected and P2P traffic still remains non localized.
- **Class 2:** There is ISP intervention in the overlay in such a way that peers continue normal operation without realizing that such interventions occur. This can be reached via the use of proxies that can affect the control plane with the redirection of content requests to local peers, or at the data plane with content caches which act as normal peers and are strategically placed in the network. These methods are advantageous because they do not require any changes to the P2P protocols, because the ISP has an active role in molding to the overlay, intercept traffic, and either help or guide it in a way that favours them. Indeed, these techniques haven been proven to work , as concluded in [27], and put into practice, for example, in [47] and [48] via the specification of a BitTorrent tracker that is programmable to allow for P2P qualitative differentiation and ISP-cooperative traffic engineering that could help reduce inter-domain traffic significantly. Additionally, in [49], with the injection of special nodes on the Gnutella overlay which interface with the base protocol nodes but with the added caching and load-balancing mechanisms, in the attempts to alleviate the great amount of "free riding" that exists in Gnutella applications - as discussed in Section 2.1.3 - by minimizing the total amount of query floods and more evenly distributing content throughout the network for increased infrastructure resourcefulness. However, this class of mechanisms is not without its challenges - firstly, it involves much effort by ISPs, as it requires structural upgrades and constant adaptiveness to new and changing P2P protocols. Perhaps worse, even when considering proper budget and maintenance, such methods can prove themselves to be not possible at all - for legal reasons, as data caches could possibly contain illegal content; and

for technical reasons, since the packet inspection required by ISPs to detect and steer P2P traffic may be blocked due to the peer's attempts to mask its traffic. It's also important to note that since no application-layer input exists, this approach could be one-sided in the sense that only ISP needs are favored without directly considering application needs.

- **Class 3:** Relative to previous classes, the active role is switched and it is the P2P system itself that acts in regards to the underlay it operates on, but without ISP involvement. Peers probe the neighboring network elements as a way to get more familiar with connection properties, and act on these probings during operation, e.g., when choosing neighbours to construct the overlay network with, or when choosing from whom to request a given resource. Whilst these methods can be advantageous for both applications and ISPs, it can't be assumed that to always be the case - as peers have no ISP input, they cannot have a full knowledge scope of the network and its needs, and as such these application optimizations can end up being more hurtful than helpful. For example, consider a scenario where a peer uses RTT measurements to choose between two candidate peers, but the one that is geographically closest to it belongs to another AS, and his preference for it to supply the service would incur in more infrastructural costs. The paper describes this class as a "win-not-lose" situation, meaning that while the P2P system can, in the right circumstances, improve their performance via measurement-oriented strategies, the ability to act in a way that positively affects the underlay, without any feedback from ISPs, cannot be guaranteed. Such an example of class 3 mechanisms could be seen in [23], which improved BitTorrent's download performance and even managed to reduce ISPs' backbone and cross-ISP traffic. The technique consisted of having peers send traceroute measurements to the tracker, which in turn grouped the peers into local, intra-ISP and inter-ISP groups, with the assumption that inter-ISP links generally have much more latency than the rest. As peers would later query the tracker for content, the returned peer list would be biased in such a way that promotes traffic locality. Another example of this is [50], which devised a CDN-P2P hybrid where peers utilize RTT measurements to group themselves by separate orders of geographical proximity with the same intent of the previous example, which is to localize traffic whenever possible. This technique also proved itself to be advantageous, as the solution was more efficient in terms of reduced total service disruption time when compared to a previous iteration of the hybrid

architecture which used random peer selection to look up available target peers. As a final example, [51] proposed a node binning scheme that groups nodes of similar orders of magnitude of RTT values to pre-defined landmarks, and utilized such scheme for topology-aware overlay construction mechanisms in some unstructured and structured P2P overlays. Results allowed to conclude that even surface-levels topological information is advantageous and can significantly improve application performance.

- **Class 4:** Full and active cooperation exists between the ISPs and P2P systems. The role of the ISPs is to provide information and guidance, and P2P systems let themselves be influenced during operation. It is the methodology that most comes close to a mutually advantageous scenario for both parties, given that they both keep the entire group's needs in mind. For example, [52] proposes an oracle that receives as input a list of candidate peers that the querying peer is considering connecting to, and ranks them by client connection proximity. Such method was tested in a simulated environment and proven to decrease negotiation traffic and improve scalability of P2P networks. Similarly, [48] proposes a framework for programmable trackers in a BitTorrent scenario, providing an interface to directly define tracker behavior which, given ISP input, can provide a collaborative environment between infrastructure administrators and the BitTorrent users in an attempt to, among other things, localize traffic. The functional intent of the oracle pattern is that he possesses privileged network information and acts on it to provide guidance to querying applications, and thus has the power to impose policies and optimizations unto applications, e.g., pair peers which are the least number of network hops apart via a Dijkstra algorithm using link costs derived from network-related ISP insight. Another approach that could be used by the oracle proposed in [53] contains algorithms to dictate peer selection, task assignment and rate allocation. The method requires the full network topology as input - including link capacities and peer service costs - to minimize file downloading time and cost. The oracle would also be free to enforce ISP biases as its preferences by modifying such algorithms to, for example, minimize usage of costly links (such as inter-AS ones, and subject to peering agreements). The ALTO working group - whose work this thesis attempts to materialize into a working system and further extend its features - was formed to standardize the oracle-user scenario so it could be properly used in many situations at the scale of the Internet.

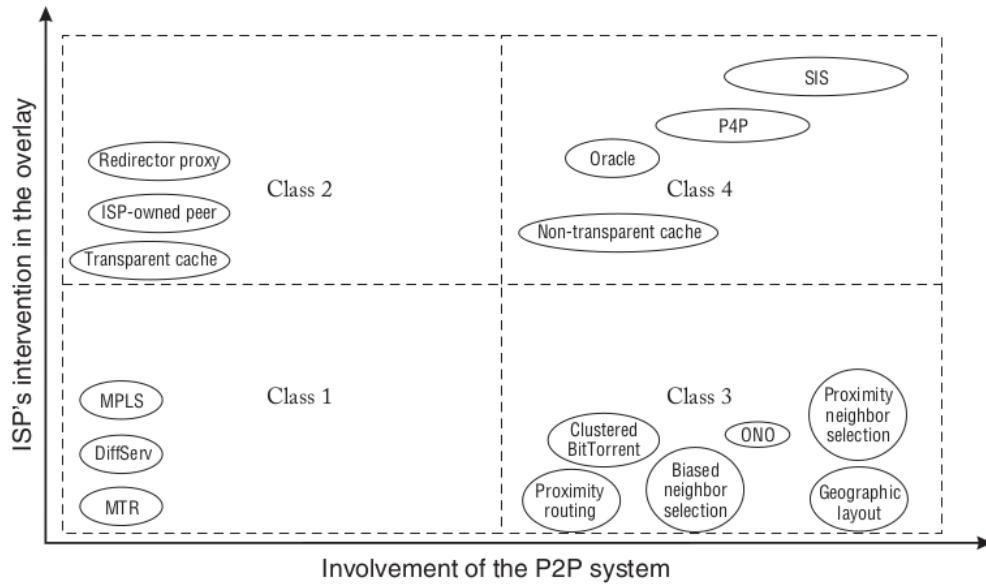


Figure 2.9: Approaches to decrease tension between P2P applications and ISPs grouped by their involvement [27]

2.4.2 Content Distribution Networks

Given the current share that CDNs have on global Internet traffic of today, coupled with the demand for a good QoE by end-users, it should come to no surprise that this application domain has also been through efforts to optimize its traffic. One such way to do so is to optimize client query redirection, i.e., better choose which edge server should be attributed to an end-user when a name resolution is requested for some content.

[54] considers a CDN built to deliver video data where some given set of content exists redundantly in many edge-servers, and presents an algorithm where the choice is made to optimize client download time, which in turn has to consider the network parameters at time of request, as well as current server load.

Some simple, flexible and scalable techniques exist that utilize no ISP input. For example, [51], mentioned in Section 2.4.1 for its P2P overlay construction with a binning technique based on RTT measurements to landmarks, also utilized such binning technique for improved server selection. Similarly, IETF tackled application traffic optimization via multi-CDN cooperation, and devised a problem statement in regards to *Content Distribution Network Interconnection (CDNI)* [55], which outlines the efforts re-

quired to specify a set of interfaces that allow for the interconnections of many CDNs, with the added benefits that a multi-CDN system, over an individualistic one, will have better properties, e.g., in regards to availability, coverage, and supported capabilities, as well as better QoE for the end user, and reduced delivery costs for the service providers. The four devised interfaces (CDNI Control interface, CDNI Request Routing interface, CDNI Metadata interface, and CDNI Logging interface) are all to be operated at the application layer, and the group states that no new application protocol needs to be devised, and instead existing ones could be leveraged, e.g., HTTP, Atom publishing protocol, *Extensible Messaging and Presence Protocol (XMPP)*, and in particular to the CDNI Request Routing interface, the ALTO protocol, who is the focus of this work, could enable CDN server footprinting.

Centralized network measurement repositories for wide consumption were tackled in project such as *Internet Distance Map Service (IDMaps)* [56] and *Global Network Positioning (GNP)* [57], that describe architectures for a global distance estimation service, leveraging measurements made by specialized nodes that retrieve raw network data, and heuristic provide scalable and functionally reliable path costs in metric such as bandwidth and latency. These consist of systems that centralize and share network probing results to querying entities, thus minimizing overhead traffic on the network. Such advantage goes outside of the CDN realm, being useful for any overlay-residing application that wants to utilize network probing to be more underlay-aware for application optimization.

[36] argues for the advantage of CDN-ISP cooperative interactions and overviews three possible strategies that will be now discussed briefly: *Provider-Aided Distance Information System (PaDIS)* [58] is a system deployed and controlled by ISPs that monitors the network by listening to *Exterior Gateway Protocol (EGP)* and IGP messages and contains a privileged view of the topology and its status. It provides a service that ranks host-client pairs in regards to, for example, delay, bandwidth, or hop count, and experimental testings concluded that the download times of content provided by CDNs that utilize PaDIS could be improved up to a factor of four, and generally gives much flexibility for ISPs to engineer traffic; *Content-Aware Traffic Engineering (CaTE)* [59] is designed in a similar manner to PaDIS but requires no client-side configuration, and experimental results concluded that network wide traffic was reduced by 15%, link utilization was reduced by 40%, and user-server performance generally increased; *Network Platform as a Service (NetPaaS)* [35] was devised to fulfill two key enablers in a fruitful CDN-ISP collaboration - user-server assignment, as it was tackled in the previous two examples, and server allocation, i.e., where should a CDN deploy its servers

and its contents. This service, besides having the advantages of increased application performance and better ISP traffic control that were also mentioned in the previous two solutions, also facilitates the task of server allocation for CDNs, reinforcing the discussed advantages and optimizing CDN operations. Still in the topic of CDN's edge server selection, [34] suggests an SDN-oriented solution that combines the performance of DNS load balancing with the low management overhead of IP anycast. Load balancing is performed at the SDN control layer by applying collaborative efforts between the CDN and the ISP. This example of layer cooperation can allow for many optimization opportunities that leverage an existing and low-maintenance mean of request routing with the flexibility of SDN solutions.

2.4.3 Server-client applications

Attempting to optimize web server selection, [60] argues that DNS-oriented solutions, which select the nearest server but also employing load balancing, may not be the best at optimizing server-client QoS levels. Instead, it proposes a selection based on QoS measurements, from which three types are distinguished: a static method, such as choices based on least number of hops to server (which is unlikely to change); a dynamic method, consisting of network probing to assert, for example, RTT values to the servers; and statistical methods, which decide based on a larger set of measurements previously made in various points in time. Utilizing the latter method, RTT measurements and web-related request benchmarking is made, such as time to establish a *Transmission Control Protocol (TCP)* connection, elapsed time from an HTTP GET method to first packet received, time to retrieve data fully, etc, every five minutes and spanning several weeks. The work concluded that statistical methods used to select between multiple equal web servers had high correlation with download time from the selected server, but optimizations should be evaluated in regards to computer workload and the amount of probing traffic.

Tackling a similar challenge, [61] proposes a method of server mirror selection which is better optimized than the more popular approach of giving the user the selecting choice. The proposed solution's architecture consists of two types of agents: a client agent, which monitors the mirror server it was deployed in and stores static information, e.g., geographical location of server and maximum capacity, and dynamic information, e.g., current load and bandwidth. This information is then sent to the other role of the architecture, the server agent, which compiles it and acts as an oracle

that is queried by users whenever mirror selection is needed, replying with a ranking of candidate servers based on bayesian networks.

Congruent to the task of optimizing network traffic with layer cooperation, [62] proposes a reconfigurable and adaptable overlay multicast system, further optimizing the multicast strategy - used for group communication as a means to reduce redundant traffic - and leveraging collaborative efforts between it and the ISPs to construct multicast distribution trees whilst integrating traffic engineering mechanisms for the task of network usage optimization.

2.4.4 Summary

Concluding, application traffic optimization does indeed to be a common concern for P2P, CDN, and Server-Client systems, as it improves application performance. Indeed, potential to improve exists if attempts are made to better comprehend current network status to aid application decisions, and so is made by realizing more about the underlay, whether by probing it, or retrieving that information from - or delegating decisions to - authoritative and generally trustworthy sources that keep both interests in mind. A fully mutual cooperative scenario seems much more efficient than one sided approaches. Considering ISPs, the ability to directly impact application behavior lets them engineer traffic at a more fine-grained level, that would be impossible without it. Considering applications, at worst, resorting to ISP guidance would help minimize the amount of redundant network overhead generated from all kinds of applications monitoring the underlay for their status, and at best will allow better insight that only the ISP itself could provide, with its privileged intimate knowledge of the infrastructure and centralized monitoring structure that gives information about historical traffic patterns. Thus, a win-win scenario is theoretically possible, a result much valuable assuming an existing cooperative infrastructure and voluntary participation from both layers.

2.5 APPLICATION-LAYER TRAFFIC OPTIMIZATION (ALTO) WORKING GROUP

2.5.1 Context and Motivation

Acting on research indications that improved peer selection algorithms based on ISP-provided information could help reduce infrastructural costs and increase P2P application performance, the IETF devised working groups to explore possible standardization in the area of layer-cooperation [3]. Among them is the ALTO working group, whose domain is traffic localization.

The ALTO [63] working group designed an HTTP-based protocol whose function is to allow hosts to query privileged servers on network information. The IETF-devised working group's project has gathered much academic interest, e.g., [3], [58], [62], [20], and [27], as well being suggested as an appropriate framework to help fix various problems, e.g., [59], [58], [34], and [55], and these form a subset of a larger current preoccupation with the underlay-overlay tussle and the attempt to find means of collaborative layer effort. The envisioned scenario of the service provided by the ALTO architecture, as can be seen in Figure 2.10, considers both the physical and application domains - the underlay and overlay, respectively. The ALTO service is provided by some oracle, which in turn needs to be supplied with network information that can take many forms - topological structure, routing costs, static policies, etc. - and, most importantly, such data is to be fed by an ISP or such other authoritative entity that contains truthful and relevant network information that the oracle could deem useful in aiding its clients. Using Figure 2.10 as an example, consider that "Peer 2" wishes to retrieve a given resource with the application, and after probing the overlay network - via querying a tracker, using a flood of peer pings, or some of the means utilized by structured P2P networks - the peer locates "Peer 1" and "Peer 3" as possible candidates to serve the content it wants to retrieve. Aware of the fact that choosing whom to consume a service from has impacts on both application performance and network resource utilization, "Peer 2" is to use the ALTO service, querying the oracle on information pertaining to the candidate peers, and in regards to metrics that better fit the needs of the application - because different applications could have different QoS metric priorities in mind, like a media stream with low delay needs or a file sharing application with focus on high bandwidth availability. The ISP is then be in full control of engineering how the traffic from this resource transfer will flow, and can

steer "Peer 2" in favoring "Peer 3" - since they reside in the same physical network domain, this would improve infrastructure resourcefulness and there would be no need to make use of peering links that interface with external regions. As could be deduced from this and similar scenarios, an architecture containing one or more servers that are knowledgeable on the network they reside on could be an important tool to make P2P applications locality-aware, a common goal for the underlay and overlay parties since it is a win-win scenario.

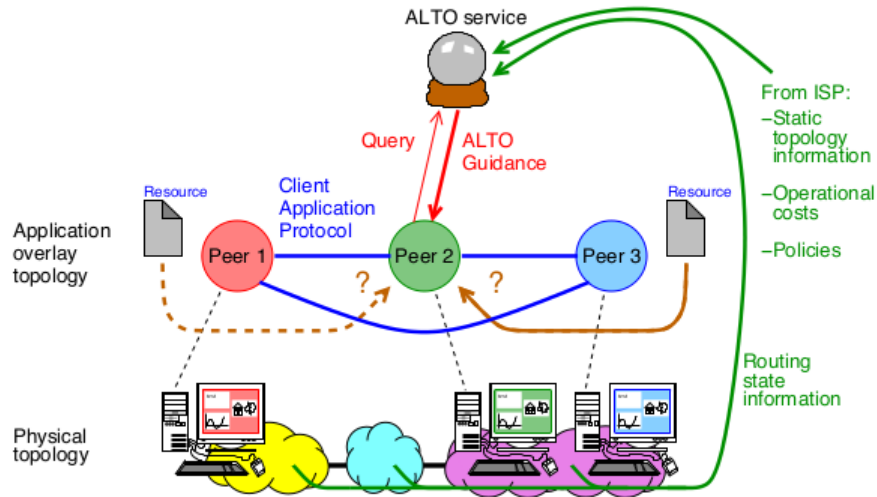


Figure 2.10: ALTO scenario [3]

Despite its origins lying in the efforts to localize traffic in P2P applications, the ALTO protocol and its encompassing system is now being considered in other fields, to be now further discussed.

A first area of interest is CDNs, most specifically the on-going works in extending the base protocol to implement the CDNI Request Routing Footprint & Capabilities Advertisement interface [64], which is a subset of the CDNI standard [55] that aims to allow upstream CDNs to query known downstream cdn for their willingness to accept content requests on their behalf. In particular, one of the main functionalities of the CDNI request routing interface is the ability for upstream CDNs to retrieve static or dynamic information on download CDNs (resources, footprint, load), which they provide themselves, and that allow the upstream CDNs to better choose the appropriate edge server that could serve a given end-user. ALTO serves as a good protocol to implement such functionality because it fits its use-case: some node - in the upstream CDN, where the content query originates - wishes to improve its routing - in regards to resolving content requests - by using information that is hard to independently de-

duce to properly choose the most efficient node - the downstream CDNs where the content resides. At a more abstract level, this is similar to the use case discussed above where P2P required assistance to more optimally select peer connections.

Edge computing, similarly to CDNs, uses a paradigm of flexible service distribution that enables deployment closer to the end user for better performance, and thus is inherently effected by network status. Current work is being made on how ALTO can be leveraged to aid the task of deployment of functions or applications in the network edge [65]. Much like the previous example, ALTO is being used to guide an application in a decision with impacts on both layers - by querying the ALTO server, the client can retrieve information that regards to *Points of Presence (PoP)* where functions/applications can be deployed, such as cloud computing provider's available resources, e.g., *Central Processing Unit (CPU)*, *Random-Access Memory (RAM)*, or storage, but also network information that pertains to the outside of the PoP, mainly network connectivity metrics, e.g., end-to-end bandwidth and delay, and routing costs. The utilization of the ALTO protocol in this context would allow edge service clients and providers, as well as ISPs, the ability to combine efforts as a means to optimize edge computing deployment that considers the current network status, and doing so would thus result in benefits for both end-users and infrastructure maintainers.

More broadly, current work is also being done in specifying abstract network entities path arrays between points [66] and time-sensitive cost values [67], both of which share higher insight of the network, at the discretion of the ISP, as a means to provide even more context to applications about the infrastructure, such as identifying potential path bottlenecks and high traffic time peaks, respectively, and thus improve the ability of applications to optimally generate application traffic on the network.

A mode of operation where applications no longer operate in disregard of the network infrastructure they run on, but instead in deep consideration of it, could help significantly alleviate the issues emerging from the tension between the underlay and overlay, and is of mutual interest - improving application performance and reducing infrastructural costs. Enabling a communication channel can thus allow for many different co-operational use cases besides the aforementioned ones - for example, redirecting users to nearby data caches or warning them of server maintenance ahead of time. The existence of an all-encompassing oracle could also prove beneficial for applications which utilize periodic network probing to guide their choices, as such information could be measured by a select few nodes in the network and applicable to all nodes which are close-by to such node in ways that the ISP deems advantageous - such as belonging to the same AS or geographically close by - thus minimizing the amount

of otherwise redundant probing required by all application entities that wanted some network status information. ISP-guided counseling, however, does more than centralize measurements that could be done by the application itself - besides also containing measurements that could only be retrieved by the ISP itself due to its privileged access to the network - such as IGP packet inspections or secure *Simple Network Management Protocol (SNMP)* queries - by handing over the decision-making process to the service provider entity, it is given power to better steer traffic in a way that favors internal policies and statuses, such as peering agreements, current traffic flow of other applications, known bottlenecks, etc., that could not be deduced by the applications alone. Thus, in the decision of how to generate application traffic, the responsibility should reside in both the application and the infrastructure as a way to benefit all relevant parties - the end users, the application stakeholders, and the service providers - and the ALTO protocol is an enabler of a mutually cooperative layer interaction system that, by becoming the standard, would aid towards a sustainable life-cycle of the Internet.

Finally, standardizing an architecture and related protocols for a clear problem domain could help a large subset of similar issues - a well defined and tested specification would exist, thus allowing many applications to leverage the ALTO protocol's functionalities to their needs, not requiring further cycles of development for a specification when one already exists. Also, the attempt to standardize the oracle pattern is helpful as it joins forces from many different domains which share common problems (many of which were exemplified previously) into a single specification that interfaces with ISP knowledge, and would target issues such as security and scalability, creating a single point of convergence that is mature enough to be adopted by with confidence, accelerating the transformation of the Internet as individual players would not need to develop their own specifications.

2.5.2 Architecture

The high-level conceptual ALTO architecture can be seen in Figure 2.11. Central to the operation is the ALTO server, which stores network information and provides it to querying clients. Such network information is provided by trustworthy and relevant entities, as could be derived by routing protocols, ISP-specific policies, historic measurements, and feedback provided by third parties regarding application performance on the network. Two protocols can be seen as part of the general architecture: the provisioning protocol - which is not currently contemplated by the ALTO working group

- should specify how information is provided to the ALTO server; the ALTO protocol
- main focus of the working group with the same name - specifies server-client interactions as a request-response interface for retrieval of network attributes. The ALTO client is the main consumer of the ALTO service, and it queries the ALTO server on network information whenever it deems such data as necessary to what it's doing at a given moment, with some potential use cases discussed previously. An ALTO client could be seen as any entity which is able to interface with the ALTO protocol with the role of a client, and as such is not tied to a specific implementation - in the example of P2P file sharing, a peer can act as an ALTO client (like the example scenario in Figure 2.10), but a tracker could instead take that role, enhancing its ability of assisting peer communication by having an embedded ALTO client that would then act on behalf of peers when querying for ISP insight as to provide an optimal peer pairing. Using the tracker-oriented ALTO client approach would minimize needed P2P client protocol modifications and thus facilitate integration with currently existing applications.

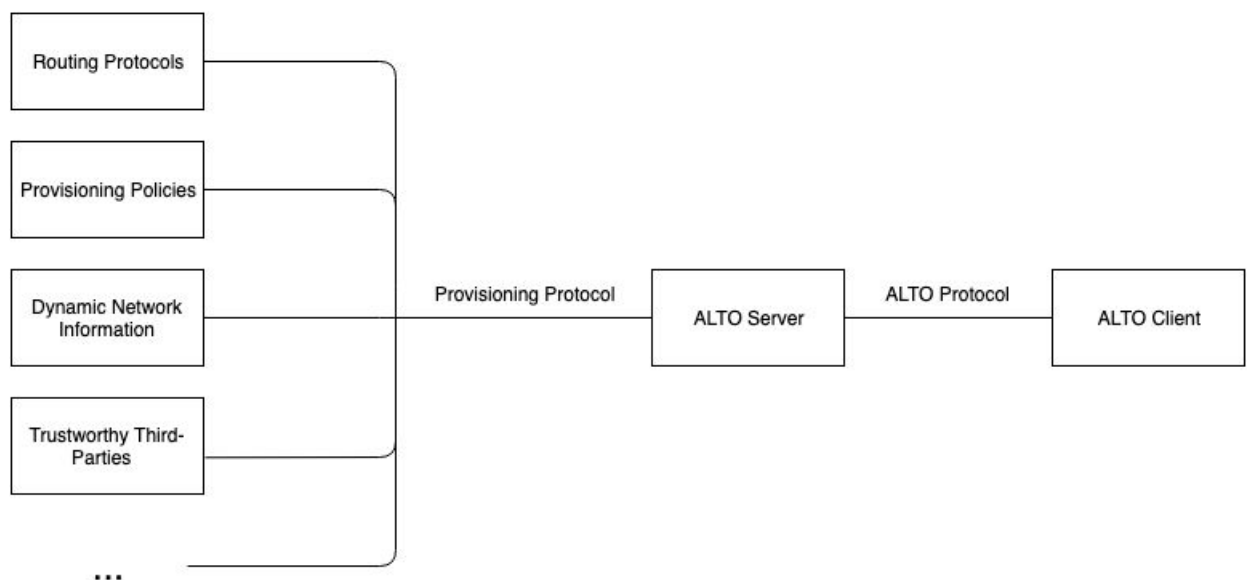


Figure 2.11: ALTO architecture (adapted from [68])

The ALTO services contemplated by the working group can be visualized in Figure 2.12.

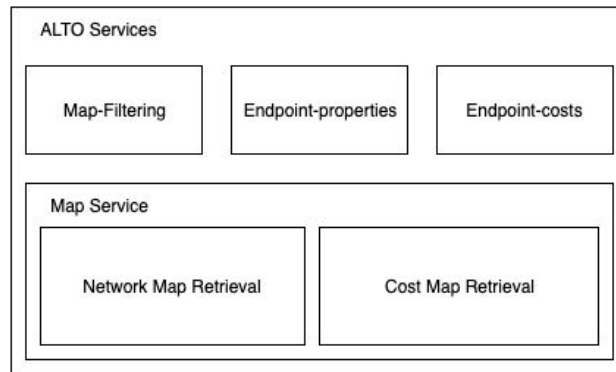


Figure 2.12: ALTO services (adapted from [68])

The ALTO server stores and provides special mappings in the form of network and cost maps.

A network map provides network location grouping identifiers and the corresponding aggregated endpoints. It utilizes *Provider-Defined Identifiers (PIDs)* as a key, and the mapping itself is left to the responsibility of the providers - it is a way of indicating that many endpoints should be handled similarly. A provider can then aggregate endpoints by geographical proximity, one or many subnets, one or many ASs, etc., and attribute properties to the aggregate, instead of the endpoint. This is advantageous not only for scalability reasons - since it can compress information - but also because it allows ISPs to abstract network endpoints into groups, thus ensuring privacy of network topology details whilst maintaining useful network guidance, as the ISP has full control of how endpoints are aggregated, and consequentially how traffic is engineered since this changes how clients interpret resources.

A second resource type provided is a cost map, which can be defined as a matrix M , where M_{ij} - with i and j being the source and destination PIDs, respectively - is the associated path cost between the two indexes. The cost has two components: its metric and mode. The ALTO base protocol only defines a single, generic, cost metric called "routingcost". However, [?] is currently specifying more concrete metrics, with many associated with QoS evaluation, e.g., one way and round trip delay, packet loss and throughput. The other cost property, cost mode, can either specify that the metric is to be interpreted as a numerical value or as an ordinal ranking among all other costs in that cost map - this is useful in cases where too much network information is not deemed reasonable to share, and a simple order of preference that doesn't expose excessive infrastructural details can suffice. The decision to separate network and cost map information into two types of resources comes from the reasoning that

network mappings are unlikely to change, whereas cost mappings could be periodically updated. As such, it alleviates client applications from the need to retrieve redundant information, and the ability to only retrieve a subset of it - this ability is further expanded in the map filtering service, which allows an ALTO client to further specify which regions of the requesting maps it wishes to retrieve (much like a "SELECT" statement from an *Structured Query Language (SQL)* database), and only these are transmitted.

Finally, the last two services focus on mappings that regard to specific endpoints, instead of abstract mappings that utilize PIDs. An endpoint is currently identified by one of the following: IP address, *Media Access Control (MAC)* address, or generic overlay ID. The endpoint property service maps to an endpoint a set of properties, e.g., geographical location or connectivity type, and the endpoint cost map has the same meaning of a cost map, but mapping to particular endpoints addresses and not abstract collections. The ISP has thus the ability to work with abstract aggregates or specific endpoints, showing as little or as much network information as it deems fit.

As could be seen, the ALTO project specifies an architecture for sharing of network-related information, with well defined roles and a request-response protocol to fulfill interactions between them. It also attempts to standardize such interactions in the form of data structures with well defined attributes which are then to be manipulated for each use case. This could then serve as a useful service for any application that wishes to retrieve network information as a means to improve its decision making at the application level. It is important to note that there are restrictions to what kinds of information are contemplated by the ALTO protocol - for example, transport-level congestion is beyond its scope, and thus should not replace conventional mechanisms. The type of data which is valid to consider, according to [69], should not be easily obtainable by the clients themselves - such as immediate end-to-end delay - and should be variable on a longer timescale than the instantaneous kinds that are seen on, for example, congestion control mechanisms, as the frequently resulting querying traffic would be counterproductive to the task of traffic optimization. Potentially valuable information that is in the ALTO scope would then have to be harder to obtain without aid of this service, and not highly mutable through time - for example, routing costs, geographical locations, network proximity, operator's policies, scheduled down-times, historical application feedback, etc.

This project is, at time of writing, still on-working, with many drafts being created and updated as the ALTO project matures and increases its domain applicability. These are, however, relating to service extension and deployment, since the main ar-

chitecture, protocol design, implementation guidelines and security analysis are fully published into their respective *Request for Comments (RFC)* documents, serving as pillars for this work, and the ongoing efforts will serve as inspirations for potential extensibility.

2.5.3 Viability

2.5.3.1 Security

Given the nature of this system - the trading of sensitive network structure information that can alter application behavior - it is quite apparent that its design and implementation are not without challenges from a security perspective. Indeed, the working group published discussions regarding security preoccupations at the development and deployment stages of the ALTO system [69] [68] [70].

Utilizing the "STRIDE" threat model [71], the main threats to the ALTO architecture can be summarized as follows:

- Spoofing of a legitimate ALTO server that would mislead clients with wrong information - this could give the malicious party the ability to change traffic to its will. Spoofing of the clients themselves can also occur, and could allow a malicious party to retrieve sensitive network data outside their permission. Finally, spoofing of a provider of network status that could feed information into the server to be spread to applications, possibly misleading them in the same way an ALTO server spoofing could, but by proxy.
- Tampering of data to mislead either ALTO servers or clients. If some unauthorized and malicious party can retrieve data that is in transit or storage and tamper with it, clients would act on information that they assume is trustworthy but in fact has been modified. As such, clients could be redirected to wrong addresses, or receive incomplete or incorrect data that results in bad decision making. On the other hand, data tampering that occurs between data providers and the ALTO server would give the latter, from a seemingly trustworthy party, untrustworthy data. This would result in the same issues that could arrive from spoofing threats. Tampering could also occur in input forms in the server-client or server-provider interface with potential to inject malicious code execution.
- Repudiation of being the source of some network information, whether it be by a third party that volunteered the data or the ALTO server itself, which would

make it difficult to neutralize and attribute culpability to incorrect or malicious sources, jeopardizing the legitimacy of the provided network information.

- Information disclosure in the form of ALTO resources being made available to entities that were not contemplated to access it. These resources could give malicious parties insight on network topology status as well as the ability to derive clients' network usage patterns by observing what kinds of resources they attempted to retrieve at a given moment.
- DoS of the ALTO server through request flooding beyond its capability, which would severely hinder - or even negate - its ability to serve legitimate users. By proxy, service denial of external entities can also happen through the manipulation of ALTO resources themselves - leveraging the system's potential to guide traffic, if a given resource is manipulated in such a way that unreasonably favors the preference of a specific subset of servers, these could be selected by clients in a disproportionate manner, and highly affect these servers' availability.
- Elevation of privileges that lead to obtaining or modifying more information than initially permitted, resulting in the previous threats being heightened.

Many of these threats are standard preoccupations for most computing systems and could be solved with state of the art solutions which are well proven and tested, as indeed states [68]. However, regarding threats of information disclosure - whilst they can be negated with in-transit encryption, what is done with this information the moment it reaches the client is hard to control - situations may arise when a client with proper resource permissions shares, intentionally or not, sensitive network information with other users who may or may not have proper clearance, in interactions outside the ALTO architecture. Furthermore, many authenticated clients with different permissions could share information among themselves which they retrieved legitimately, to get an illegitimate complete view of the network structure. Thus, individual clients could internally collaborate outside the system to bypass access control measures applied inside it. As such, it is firstly important for the ISP or third parties to carefully plan on what information they are comfortable with sharing, knowing that it may be susceptible to future disclosure outside the secure domain. Possible solutions to minimize these threats include:

- Reduce the granularity of the provided data. Intuitively, the less granular and precise the shared information by the ALTO server is, the less valuable the resulting application guidance will be, and thus a balance would have to be found

between layer cooperation and ISP privacy. One example is the usage of network groupings by PIDs instead of mapping information to concrete endpoints, working with network status around abstract entities. Another possible mean to reduce information granularity would be to utilize ordinal cost values, which instead of specifying a concrete metric, e.g., bandwidth in bits per second or packet loss in percentage, the server would give a relative preference rating with lower costs meaning lower preference. In both examples, the granularity of network information transmitted to the client is several levels higher in abstraction than the actual physical layer, and this could reduce the flexibility of applications to optimize traffic. However, they can still provide acceptable flexibility without impacting ISP privacy, acting as a much needed compromise.

- Work only with a small set of trustworthy clients that are to act on behalf of a larger subset of less trustworthy clients. For example, network status resources could only be provided to authorized cooperation-oriented trackers in the BitTorrent protocol, which would in turn use this information to provide customized replies to clients without the need to change the base protocol. Similarly, information relevant for user-server assignment could only be provided to authenticated CDN control nodes, who'd use among themselves a private virtual domain to share information about user-server connectivity and server status that would otherwise be inappropriate for any other type of user to retrieve. This is still, however, worthy of further threat analysis as restricted information could still leak outside of the system - beyond the means of spoofing discussed previously, seeing how a system behaves with ALTO guidance can give - albeit limited - insight into SP bias. To see this, consider how a BitTorrent peer could continuously query a tracker with carefully crafted parameters - such as source address and candidate peers - and attempt to derive information from the resulting action, or similarly how a end-user could utilize similar parameter modification to observe the edge server selection mechanism in action.
- Utilize terms of agreements that are to be enforced on every querying client, stating that network status information does not get used beyond its original purpose, prohibiting sharing. Although a potentially helpful mechanism to dissuade malicious users, it can be deemed impractical to apply, especially considering the scale at which this information could be shared. Thus, utilizing such means should be applied at a case-by-case situation and it should not replace ISP discretion and server resource maintenance to ensure a given standard.

2.5.3.2 Privacy

Privacy concerns are also very prevalent in the ALTO system, being an ubiquitous talking point in most of the working group's problem statements and protocol specifications. When an ALTO client queries a server for one or more network status resources - in the attempt to optimize the application traffic it will generate in the near future - certain parameters can be passed to the server that can make the response be more personalized and contain more granular information. For example, a real-time P2P media-streaming application seeking ALTO guidance to help choose among a list of candidate streaming peers may wish to include in its query helpful parameters such as the peer list itself, the desired QoS, and the network position of the querying client itself. Indeed, these and more patterns will help increase the effectiveness of the ALTO server's guidance in helping the client application achieve its goal, but such happens at the expense of potentially allowing an ALTO server to infer on user pattern statistics. Even assuming that the previously discussed information disclosure threats are nonexistent in the ALTO system, privacy concerns can arrive from client applications because the resource queries they need to produce can contain information about what the client either will or wants to do. This is recognized by the ALTO working group as a possible concern [68] [69]. In response, they state that the clients should firstly be cognizant about the potential tracking risk that is associated with the usage of the system and, as an attempt to make tracking harder, they could disable HTTP cookies and/or opt for more vague query parameters, e.g. by randomizing some bits on endpoint addresses or simply using more broad addresses, whilst being aware that the helpfulness of query results may vary with increased parameter obfuscation.

Very much like client privacy, ISP-related privacy is also considered by the working group. PIDs were created as a means for ISPs to abstract network components as a collection of single network endpoints with similar properties, helping them not to disseminate network information that is too sensitive, and in turn also allows clients to make queries based on these identifiers and maintain a higher level of privacy. An ongoing proposal for protocol extension includes path vectors [66], that aim to represent information on the intermediary hops from a given source-destination pair, and each of these hops is represented as an *Abstract Network Element (ANE)* that, similar to PIDs, give ISPs the ability to under or over-abstract the topological representation that gets published to clients, giving more options to balance guidance usefulness with provider privacy. Other solutions could also be considered depending on the needs of the clients and the direction of the project as a whole. For example, the

servers themselves could operate on a secure communications channel and maintain a clear agreement on what can and cannot be made with the collected information. Alternatively, clients that wish not to impose much trust on the server's claims not to track them could make bulk queries (or use proxies to do so for them) and privately filter out the relevant information, heavily restricting on the ability to retrieve user activity patterns.

2.5.3.3 *Incentivisation*

Incentivization relates to creating and divulging, to both layers, incentives to a fully cooperative layer relationship that is inherent in the oracle pattern adopted by the ALTO system. It is quite the challenge to fundamentally change how applications behave on the internet, as indeed is to ask of ISPs to launch a view of their infrastructure to the outside world. [27] notes incentivisation as one of the key challenges in overlay-underlay cooperation in regards to P2P applications, stating that incentive mechanisms need to exist to ensure that both layers agree to participate in, and maintain, a cooperative relationship. According to the ALTO problem statement [69], the incentives for both parties to act on the system are the advantages that derive from using it - clients are to expect better application performance by leveraging ALTO guidance, and similarly ISPs should expect that their internal goals, such as an optimization of infrastructure utilization, can be met with the increased traffic engineering ability that results from their oracle role. If the overlay consuming ALTO guidance has a manageable number of accountable entities - such as a single CDN or data center that the ISP agrees to partner with - it is realistic to maintain a cooperative agreement that can be solidified with feedback and service agreements. However, if the overlay utilizing the ALTO system makes it hard to pinpoint accountability, such as a large P2P application with many users, it will naturally be harder to ensure that the power dynamic between layers doesn't shift beyond an equilibrium. In these cases, policies could be created and enforced to give insurance to both parties that a cooperative relationship is maintained.

The ISPs could too, like mentioned above, lack proper cooperation, as they are found in a new power dynamic that would leverage their application traffic engineering capabilities to steer traffic in a way that is advantageous to only them, or at least in a disproportionate manner. Again, much like the lack of cooperation by clients, it is difficult to guarantee an equilibrium in the power dynamic between layers, but by guaranteeing improved QoS levels for applications that utilize ALTO cooperation, ISPs

become responsible for guaranteeing that these improvements are met, fearing client abandonment otherwise. Giving freedom to both layers on how they act ensures that the system evolves to a common ground that benefits both sides, at least enough to justify them remaining there.

Finally, if the application-ISP tussle becomes harsher and unmaintainable - a point that was argued for in the network infrastructure effects portion in Section 2 - a co-operative system such as ALTO may become necessary - and thus beyond preferable - meaning that ISPs may be forced to block or throttle traffic that it cannot route properly, as it historically happened. Thus, acting with ALTO could go beyond a voluntary action into a symbiotic necessity, meaning that both parties have to act cooperatively to maintain network sustainability. Regardless, the best approach seems to be that the system must in of itself be self-justifiable, meaning that the advantages that it brings should be enough to convince both parties to act on it. ISPs are nevertheless free to deploy their own incentive mechanisms to facilitate early application adoption, that could include monetary rewards or routing privileges, but doing so could damage network neutrality.

2.5.3.4 *Network Neutrality*

As stated by [72]: "According to most network neutrality proponents, network neutrality rules are intended to preserve the Internet's ability to serve as an open, general-purpose infrastructure that provides value to society over time in various economic and non-economic ways. In particular, network neutrality rules aim to foster innovation in applications, protect users' ability to choose how they want to use the network, without interference from network providers, and preserve the Internet's ability to improve democratic discourse, facilitate political organization and action and to provide a decentralized environment for social, cultural and political interaction in which anyone can participate.". Network neutrality has been a popular point of discussion as society grows around the Internet, sparking debates around the world on what the best course of action should be - for example, regulations were introduced by the *Federal Communications Commission (FCC)* [73] in the United States to police network neutrality [73], and the European Union has a framework for net neutrality laid down in Article 3 [74]. However, potential violators of the spirit of a network neutrality exist, such as British Plusnet's [75] usage of *Deep Packet Inspection (DPI)* to implement limits and differential charges for different traffic [76], or Portuguese MEO's [77] smartphone

contracts which include zero rating programs for a given set of services [78] that bundle applications such as Facebook [79] or Spotify [80].

Network neutrality advocates are concerned with guaranteeing that ISPs keep Internet communications free and do not discriminate based on the traffic's specifics, such as platforms, applications, or source and destination addresses. On the other hand, opponents of net neutrality, among them the ISPs themselves, broadband and telecom companies, and hardware manufactures, argue against net neutrality - they claim that it would reduce incentive to invest, as investments would be harder to insure without the ability to charge higher rates for better infrastructure capabilities. Zero rating programs, such as Wikipedia Zero [81], which provide Wikipedia [82] pages with no charge to a select group of low income regions, are popular in developing countries [83], provide to select regions Internet content they could not otherwise get, but in the form of a non-neutral view to the network. Additionally, with net neutrality, the ISP's ability to route traffic could itself be at jeopardy - as [84] states in its solution to compromise net neutrality with QoS demands via service differentiation, the Internet is growing at an astonishing rate, as are the demands of applications, and operating the infrastructure on a purely best-effort basis will not be sufficient without a constant provisioning of such infrastructure to keep up with demand, and this too may not be viable nor even possible. Thus, discriminating by traffic services may be needed to guarantee that, say, real-time medical information gets priority over real-time media streaming, which in turn gets priority over e-mail or file sharing.

Considering that the ALTO system behaves in an oracle pattern of cooperation where a single entity - the ISP - is able to heavily influence the traffic patterns of the applications it aids, on the promise of a cooperative network underlay-overlay relationship, such system could violate the principles of net neutrality. In particular, this could happen if the oracle either blocks, or at the very least provides different guidance to different clients, depending on where the query originated from - e.g., which application, which source address, or other defining characteristics. A possible consequence of such a system guiding the Internet could be that given applications can consistently have better QoS measurements not on the basis of the application's implementation, but on the ISP personal biases. Oracle systems such as ALTO do not seem to be analogous to other traffic engineering strategies, such as the usage of MPLS, DiffServ, nor to other means of ISP intervention on overlays, such as the deployment of data caches and redirector proxies - this is because the oracle system, in contrast to the previously mentioned strategies, is one of mutual voluntary and cooperative nature between ISPs and applications. However, it could be argued that

if the ALTO system offered guidance to applications in such a way that consistently resulted in better application performance, such applications would be pressured to use such guidance as a means to remain competitively viable, and the ISPs would then have a platform to influence a considerable amount of traffic to their will, being in a position to, depending on how they treat guidance requests, break network neutrality. This neutrality concern can be alleviated if application guidance operated on classes of traffic, e.g. real-time communication or file sharing, thus operating on traffic aggregates to insure QoS levels needed to given application types, but never discriminating beyond such given classes.

As the protocol is defined [68], the provided network status information is truthful and guidance is optional, and neutrality can then still remain outside of the system, since no routing measurements exist within it. If particular implementations of the ALTO system give guidance in such a way to guide traffic in a discriminatory fashion, and if such guidance has advantages that much outweigh any alternative - thus rendering it beyond optional - a case can be made for how ALTO as a concept can break network neutrality - for all its advantages and disadvantages discussed below - as the ISP can utilize discriminatory behaviour to treat applications on their infrastructure differently.

2.5.3.5 *Multi-Domain orchestration*

The Internet as we know it today spans the entire globe and is rather complex in nature. According to [27], the classic vision of the Internet consisting of a network of transit and stub ASs no longer seems accurate, as it now is much more complex - for starters, the role of network owner and service provider are separating, and Internet access is provided by numerous competing ISPs. As a demonstration of such complexity, Figure 2.13 displays how the Internet is structured into many tiers of different service providers. It can thus be inferred that the act of layer cooperation can get harder when the influence domain increases and potentially spans many different ISP regions which will inevitably act differently as they can have different technologies, biases, policies, and overall goals. These per-ISP biases can make it difficult to guarantee that traffic optimization spanning multiple administrative domains is actually useful and achieves the cooperative nature in mind - for example, an ISP may not be comfortable categorising end-point costs of a given metric, thus making path calculations that pass through that ISP domain not viable. Regardless, per-domain ISP guidance has nonetheless plenty of potential - for example, the ability to localize traffic, as entities

outside of domain can be identified by ISP, and similarly per-domain optimization of resources can still be useful when such domain is large, and can be applied to high-volume operations such as those in a data center. The ALTO server within a given domain can also leverage probing measurements and feedback statistics to derive information in areas whose topological detail is unknown, giving a partial network view that contains topological insight and also information derivation that, whilst not being as good as a complete topological insight, may nonetheless power a cooperative effort within a given domain with good results. Some data may, however, be both not shared by an external domain nor derivable - endpoint property information, such as network connection types, or server footprints - e.g., available CPU, RAM and storage - can only be retrieved by authoritative entities in a given domain and probing solutions may not be available, thus considerably limiting the applicability of a single ALTO domain.

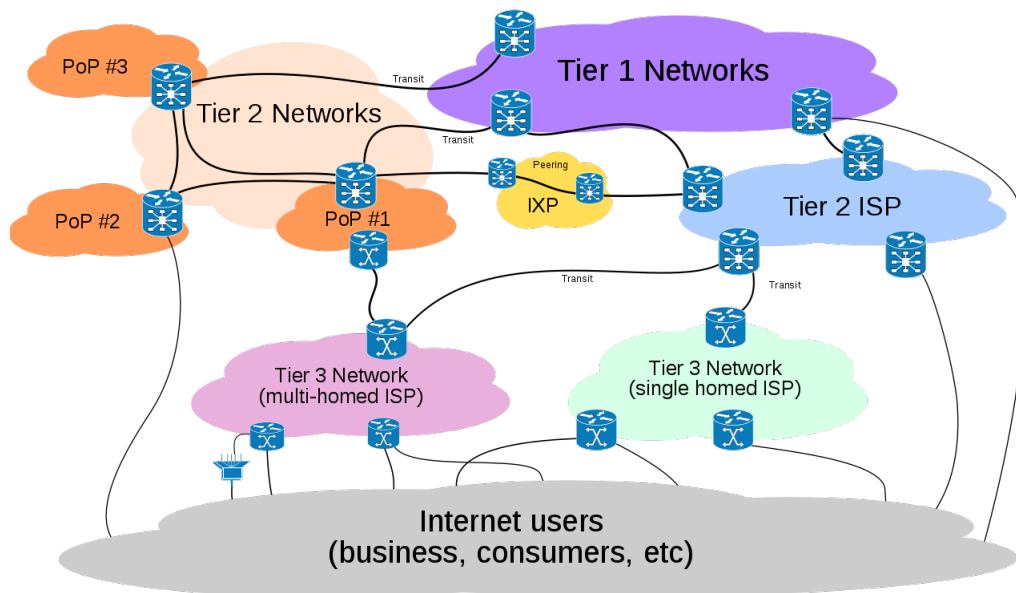


Figure 2.13: Conceptual representation of ISP diversity on the Internet

Even assuming that all ISPs are comfortable with sharing sufficient information, ambiguity may arise - considering a cost map with the generic "routingcost" cost metric, ISPs could internally calculate routing costs differently, and prioritizing different goals, e.g., reducing overall link usage versus reducing inter-AS traffic first and foremost. The base ALTO protocol specification states that each network region can provide its ALTO services, which in turn convey network information from their perspective. A network

region, per the protocol specification, consists of a given administrative domain, such as an AS, an ISP, or a given set of agreeing ISPs [68], thus implying that if multiple ISPs share an ALTO server they must reach a consensus on what network status is available for query from the outside. Furthermore, the ALTO working group's deployment considerations [70] document states that an ALTO client can query a single server for one or many metrics, or he can additionally query multiple server instances on different networks [70]. It is explicitly stated in the document that each server could give guidance for only a given network partition, and such guidance may wildly differ between them due to the fact that different algorithms and objectives may have been applied. The document also states that, in regards to extending the reachability of a single server, three different strategies could be applied:

- **Authoritative Servers:** A given set of servers can provide guidance for all kinds of destinations to all ALTO clients.
- **Cascaded Servers:** An ALTO server can possess an embedded ALTO client and query other neighbouring servers if it cannot serve the original request, acting as a middleman between the client and the more appropriate servers.
- **Inter-server Synchronization:** Different ALTO servers communicate among themselves to expand the knowledge space.

The last strategy is still being subject to development and standardization by the working group as part of a bigger attempt to link different network regions and technologies into a single, homogeneous abstraction of the Internet. Current efforts in multi domain orchestration and relevant use case examples are summarized in the ongoing work of [85].

2.6 SUMMARY

In the taken case studies seen in this chapter, it can be clearly seen that there is room for improvement in application-layer traffic generation that can benefit both the applications themselves and the infrastructure administrators that support them. Applications struggle to achieve optimal network resource utilization, whether that be in the task of matching peers in overlay networks, deploying and attributing edge server to media clients, selecting mirror servers, etc., and solutions are being continuously

proposed and created that attempt to optimize these decisions. Traffic optimization solutions vary between them with the range of control that is given to the overlay and underlay parties. It seems to be the case that one-sided solutions can hurt the other layer in a worse case scenario, and be lacking maximum efficiency at the best. ALTO's proposal seems to bridge the best of both types of proposals that are either underlay or overlay-centric, standardizing a system and associated protocol whose purpose is to achieve layer cooperation so proper network utilization is possible. Despite many of its associated challenges - namely in the regions of security, privacy, and incentivisation - the project certainly has the potential for a more resourceful Internet that can be more sustainable.

3 | SYSTEM ARCHITECTURE AND DEVELOPED MECHANISMS

As the main proposed goal of this work is the implementation of a system that complies with the ALTO working group's devised protocol, this chapter exhibits the planned software specifications needed to implement the system as a whole, with the aforementioned protocol being a crucial part of client-server resource exchange.



Initial attention is given to the general architecture on the first section, with the goal of identifying key entities, their purpose, and how they interact among themselves. The following section will target the specification of ALTO resources, which can be considered the driving force behind the system, as they are what the client entities seek, and likewise what the ISPs wish to provide. The next section will focus on specifying the task of network status provisioning to an appropriate ALTO server, in such a way that a common interface exists among all entities that are able to increase the server's knowledge of the network's physical topology. Upon specified the way that network information is provided, the next section details how a given actor, such as an ISP administrator, can pre-process such information before it is forwarded to the server and available to clients. Finally, the task of multi-domain ALTO server synchronization and communication is specified in the form of required protocol extensions and needed mechanisms that allows the increase of a single ALTO server's knowledge space.

3.1 GENERAL ARCHITECTURE

Figure 3.1 presents a high-level conceptual model of how the network information flows in a given ISP. Network data originates in the topology itself, and is gathered into a network information aggregator by the appropriate means - this aggregator defines an interface through which network data can be uploaded, and entities utilize it to provide the network data they have collected. These entities will use different means to gather information, as the Internet is supported by a massive variety of protocols and standards for network and resource information querying. For example, a node could deploy a daemon listening for *Open Shortest Path First (OSPF)* protocol

packets to gather path cost information, and another using SNMP to gather node property information. Obviously, since the interface simply defines how raw data must be formatted to be accepted by the network information aggregator, means through which the data is uploaded are left to the source itself, and because of this static data uploads that were previously collected could be used instead of dynamic retrievals - for example, the network administrator could use previously collected information that resides in a database and upload it as is. The network information aggregator serves as a hub for network administrators to process the raw network data that was collected by the previous tier, and transform it into ALTO resources ready to be accepted and distributed by the corresponding server. This task of network information processing is where ISP policies and preferences are injected via, for example, the abstraction of network entities with the aggregation of network addresses into PIDs, and the creation of cost maps which result from the transformation of network link information mixed with given ISP goals. If, say, the administrator wished to provide a cost map between network entities which aimed to reduce inter-network traffic, it would firstly aggregate endpoints into abstract entities with common properties, as an attempt not to share too much infrastructural information, and then use the previously collected network link information, attribute higher costs to undesired links, and transform it utilizing the Dijkstra's algorithm to create a shortest path map that is bound to provider preferences. Such map is then parsed as an ALTO resource - more specifically, a cost map - and afterwards uploaded into the ALTO server with the access policies the administrator sees fit.

Table 3.1: Network node entities in the conceptual ALTO system representation

Image	Description
	Network node
	Network node participating in a given overlay network

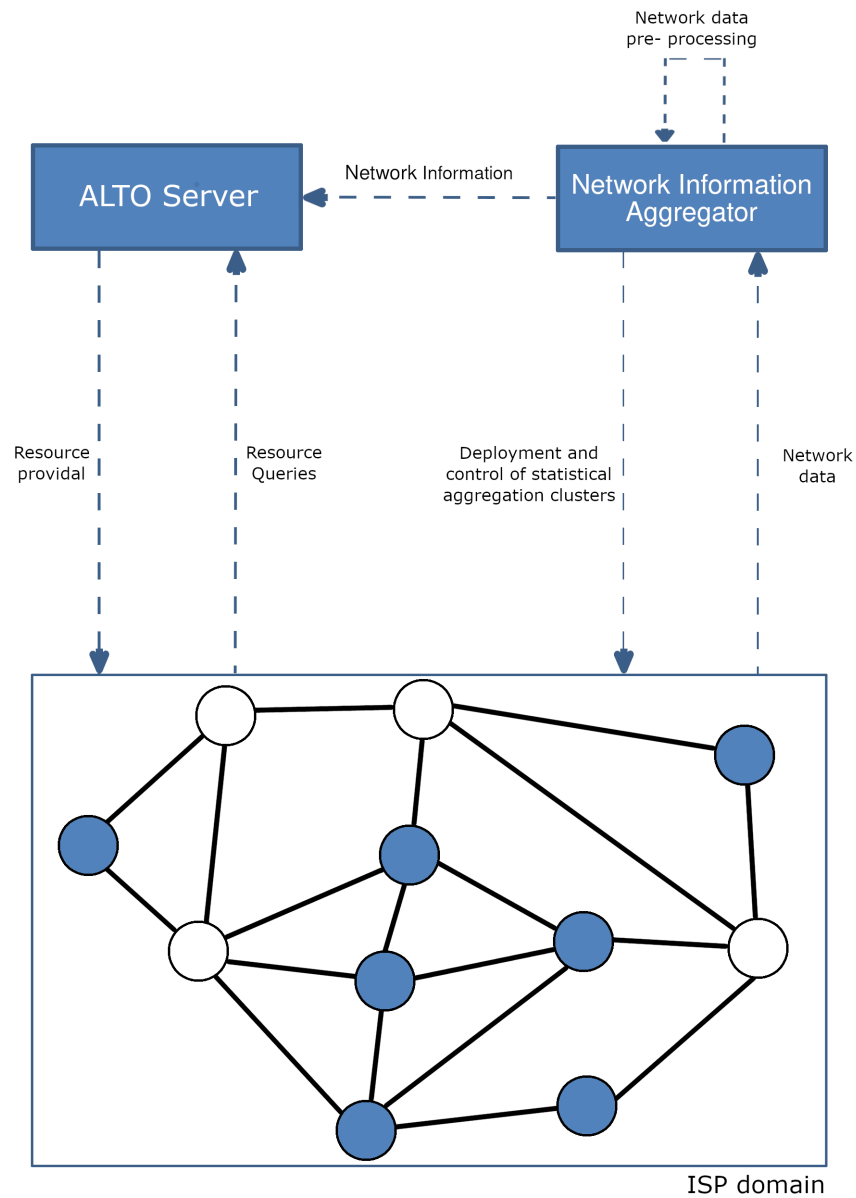


Figure 3.1: Conceptual representation of the ALTO system of a given ISP

More formally, Figure 3.2 presents the proposed system architecture. One can identify the ALTO interface - logically separated in its download and upload components - as a key factor of the system, as it allows to bridge three different application layers - the ALTO resource consumer, the ALTO resource provider, and the network information aggregator, to be further specified in the following sections.

The ALTO working group has extensively specified the ALTO protocol, which regards to resource querying, and the concrete implementation of this work will aim to comply to it. However, no resource provisioning protocol was, at time of writing, specified by the working group, nor was an interface been specified to allow network data to reach the ALTO server. It has been set as a work in progress, and the topic of network information sources was briefly discussed in [70]. The working group has grouped the tasks of raw network processing and supply into the role of the ALTO server. However, as can be seen in the proposed architecture, a different approach was taken in this work, with the roles being separated and an additional protocol proposed to bridge communication between them. This was made as an attempt to adhere to the philosophy of single responsibility, making the sole task of the ALTO server the management of ALTO resources. This aims to facilitate the independent development of the different roles, and make it easier to interchange implementations - this would make it particularly useful, for example, to deploy many ALTO servers in a cascade fashion whilst utilizing only a single network information aggregator. These are, however, only conceptually separated, and an implementation could, if it is more practical, merge the server and information provider roles into a single physical entity - which would then be similar to the architecture designed by the ALTO working group.

As most software architectures, each new communication channel represents a possible source of attack vectors and, attending to the critical security concerns posed in Section 2.5.3.1, all of these channels must be secure and reliable, as signified by the padlocks on the presented architecture. This implies that data communications within it must be block being read or altered by non authorized users, and the identity of the participating parties can be trusted and made accountable. The identified communication channels must then have methods of maintaining data integrity in transit, user authentication and authorization, and communication confidentiality.

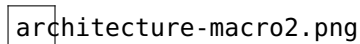


Figure 3.2: System architecture at a macro level

3.2 ROLE SYSTEM

As an access control measurement, the system will work with *Role-Based Access Control* (RBAC) methods which, as the name states, center their control policy logic around

roles, which themselves are tags that can be attributed to users. A pre-requisite is then that users attempting to access a system employing RBAC must be authenticated as a given user, and from then a list of attributed roles can be retrieved to validate if a given action is permitted according to the set rules. The ALTO resources - the main interest of this system as it is the data component requested by clients and managed by the servers - has associated to it an *Access-Control List (ACL)*, that maps, for a given set of roles, the list of user actions that are allowed to be performed to that resource, with the implicit rule that a resource's owner has full clearance. The available user actions are "read", "update" and "delete", meaning the ability to get, change the contents of, or remove the resource, respectively. This ACL must be provided by the Network Information Aggregator whenever a new resource is inserted into the ALTO server. The ISP administrator that controls the aggregator not only then designs the resource itself - adding the information that it deems important whilst not too detailed to damage privacy - but also defining access control policies on that resource, which will be then enforced by the server in future requests.

Employing access control based on roles seems appropriate for this system since roles can be applied to - and thus group - many users, and indeed that seems to be applicable on real case deployments of the ALTO system - each given application, that consists of a great number of users, can correspond to a single group, and more private scenarios, such as a data center server cluster, can also be grouped. This facilitates permission management, as the RBAC approach allows grouping of permissions into roles, which can then be quickly manipulated to affect every user associated to it - this would contrast to an approach where permissions are set per user, which would be considerably harder to manage at scale. As a user can be granted many roles, he can naturally act on the system with a role that fits the currently queried resource, if so applies, and likewise the network administrator can give permissions per role, which in turn can group as many as millions of users, or to just a single one.

An RBAC-based access control mechanism will help mitigate security threats pertaining to the ALTO working group's architecture - e.g. having unwanted users reading or tampering with data. However, for such mechanisms to be viable at all, authentication systems need to also be employed to help verify that the users are indeed who they are announcing to be. Authentication mechanisms are, regardless, of extreme importance, as they additionally help mitigate spoofing security threats. Data breaches are not, however, totally mitigated with authenticity and access control mechanisms. After an entity gets a resource and acts outside the system, it becomes out of its control and these mechanisms cannot be employed. This means that there are no guarantees

that the resources are shared outside of the system's domain and consequentially there are no security guarantees after that point. Because of this, privilege attribution by the ISP administrators not only give clearance to do a certain action, but also imply that trust exists that these users will not be improper with the given resources, such as sharing it with users with improper clearance.

Figure 3.3 provides a high-level communication diagram of how access control is enforced - the ISP administrator that uploads the resource into the ALTO server appends to it an ACL that maps actions to the considered roles, with the implied meaning that those that weren't considered have no permitted actions. When a resource consumer requests an action - which is expected to be a "read" one - and proper authentication was performed to verify its identity, the server checks that the roles associated with that consumer have the requested action allowed in the ACL and, if indeed that is the case, the action performs as expected.

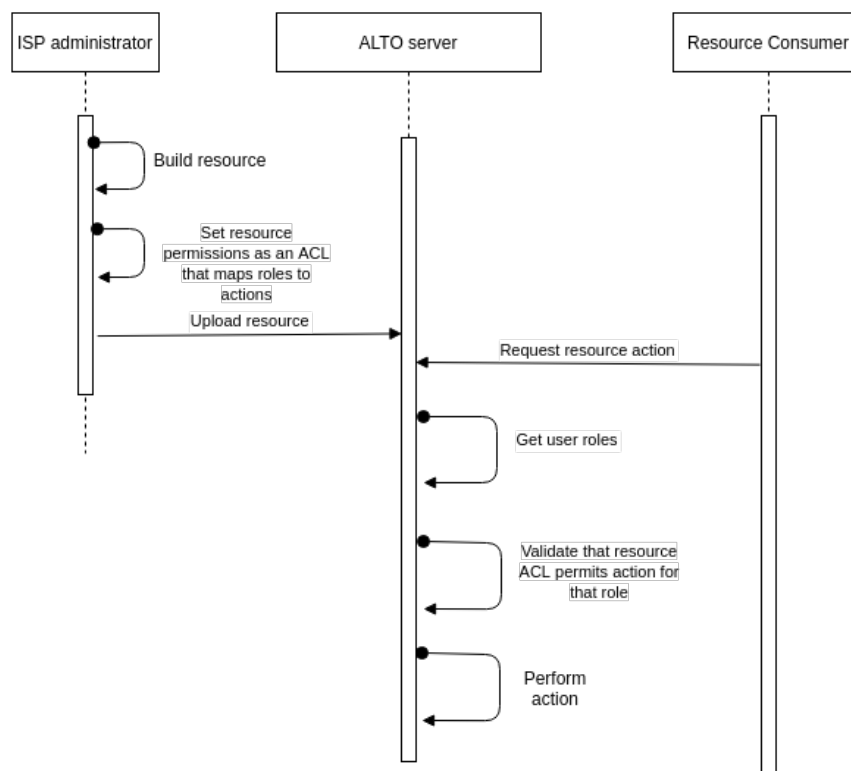


Figure 3.3: High-level communication diagram of a successful resource action request

3.3 RESOURCES

ALTO resources are pieces of network information which are provided by an ALTO server and consumed by ALTO clients that ideally would use such information to aid their application-level traffic decisions. All ALTO resources can be separated into the following components:

talk about sco

- **Meta information:** Data which regards to the resource's profile, that enable the client's ability to interpret and cross-reference the network data within. Meta information contains the resource's name, and if applicable, version, resource dependencies and cost details: enclosed cost modes, metrics, and descriptions. Finally, also belonging to the meta section of the resource's information is the resource's ACL which, to a given set of roles, specifies the allowed set of actions.
- **Network status information:** Data structures that give a characterization of the ALTO Server's vision of a network. Concretely, these can map network properties to a node - such as the connection types of their interfaces, or their geographical location - they can aggregate many network addresses to a single identifier, or they can map properties to a node link or end-to-end path - such as link or cumulative routing costs.

Meta information can be seen as a resource's header, containing data that regards to the network status and helps better handle it. Following the defined protocol [68], this field includes the resource's name for all resources, which is needed for identification and indexing, and all other fields are dependant on the type of resource: at this version of the protocol, only network maps are version-able, allowing ISPs to reference different versions of a network map as they are updated, maintaining support for previously referenced versions; cost information is, naturally, only applicable to cost maps, and gives insight on how the numeric costs are to be interpreted, i.e. what their mode and metrics are, and what description it has. Finally, extending to the protocol is the addition of an ACL as a solution to access control needs. An ACL is defined as a matrix, with each entry defining a user role and actions - discussed in Section 3.2 - as a restriction on what a given user was given clearance to do.

The network status information of a network map groups endpoint addresses into a single PID as a text literal. Akin to the working group's protocol, accepted endpoint address protocols include *Internet Protocol version 4 (ipv4)* and *Internet Protocol version 6 (ipv6)*, utilizing a 32 bit long bitmask to identify a subnetwork. Similarly, support

for aggregation of MAC addresses was added, with a 48 bit long bitmask to identify address ranges, similar to the IP variant. Additionally, generic overlay IDs can be added with the key "priv:X" - with "priv" meaning private scheme - where "X" is the qualified name - this naming scheme was adapted from the endpoint property map's specification done by the ALTO working group, for semantic consistency. As endpoint addresses utilizing this scheme aren't restricted to any type, their interpretation is also left to the client - for example, if a server defines that an endpoint addresses with "priv:my-overlay" can use regular expression to specify address ranges, a pre-agreement must exist with a client. Of course, if a given addressing scheme besides the previously mentioned ones becomes of relevant wide appeal, it could afterwards become part of the specification, but the existence of a private addressing scheme with liberal type and semantic verification gives liberties outside of the protocol for network status supply schemes that aren't supported officially. A valid network map must unambiguously map every address in the domain range to a single PID, and whenever multiple matches occur, wins the longest prefix match. As the custom addressing schemes let the network map be interpreted in an undefined way by the protocol, the server cannot properly assert to the matching validity, and thus default protocol addressing schemes for network maps should be preferred, as semantic validity in private addressing schemes is not checked. Table 3.2 provides an example network status component of a network map within the topology in Figure 3.4. Three PIDs are given, each taking portion of an ipv4, ipv6, MAC, and custom overlay address range. The private address scheme groups users in regards to their private overlay ID, and it can be seen that nodes with ID 1, 3, and 4 are grouped to a single PID, which can be seen to belong inside the ISP domain. Lastly, nodes 2 and 5 are given different PIDs as they reside outside the domain but are reachable through different peering points. The ISP could then in this case leverage the network map to logically group collections of endpoints by reachability - those local to their domain, and those reachable by one of the two possible peering points, which could be subjected to different peering agreements and as such should be treated differently in resources that reference this network map.

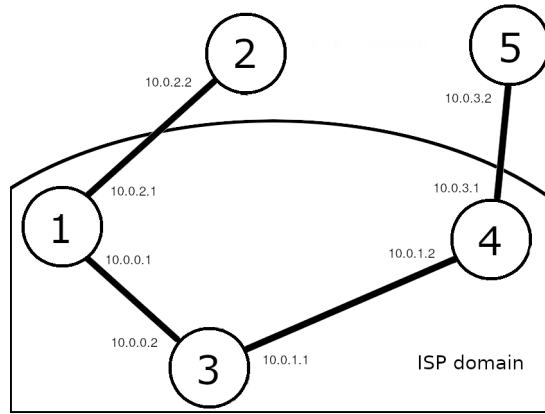


Figure 3.4: Example network topology with ISP boundary

PID	IPv4	IPv6	MAC	priv:my-overlay
1	[10.0.0.0/24,10.0.1.0/24, 10.0.2.1/32, 10.0.3.1/32]	-	Do-9F-BF-2A-00-00/32	[1, 3, 4]
2	[10.0.2.2/32]	-	Do-9F-BF-2A-FE-00/40	2
3	[10.0.3.2/32]	-	F8-BB-oB-oA-AA-AA/40	5
4	0.0.0.0/0	::/0	00-00-00-00-00-00/0	*

Table 3.2: Example network map referencing Figure 3.4

A cost map contains a list of cost map matrices, with each matrix setting pairwise values between an origin entity and a destination entity. If it is a standard cost map, these entities are represented by PIDs that can be cross-referenced from a network map which this resource depends on, whereas if it is an endpoint cost map, these entities are endpoint addresses which, similar to network maps, include ipv4, ipv6, MAC and private endpoint types. A matrix must specify the type of cost represented with both their cost type and cost mode, with available options being the ones specified in the ongoing ALTO group's cost metric specification [86]. Optionally, a cost matrix can specify calendar information about that matrix - similar to the current work in [67] - which signifies that besides having single-value costs - which are obligatory for any cost matrix - it also contains a time-sensitive list of costs that must be interpreted according to the calendar information provided, and give a chronological overview of what the costs will be in the future. If the ISP contains full topological knowledge of the resource it is sharing, the information that can be provided by the cost maps can be quite detailed. Consider the topology in Figure 3.5 with its five network nodes - from the cost map presented in Figure 3.3, one cost matrix depicts a generic "routing-

cost" cost matrix, depicting routing preference as a shortest path map with a Dijkstra algorithm and hop count as its link cost, and another could provide a "delay-ow" cost matrix, depicting expected one-way delay in milliseconds, as the cumulative calculation of known link delays.

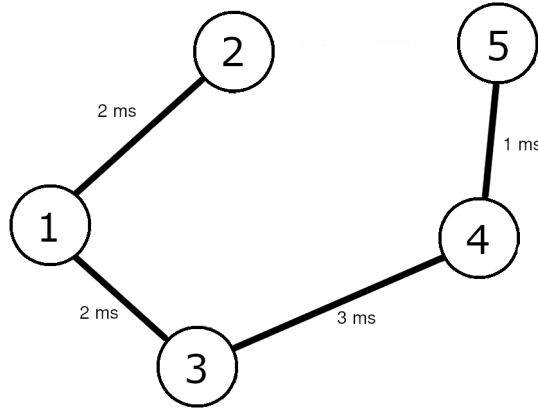


Figure 3.5: Example overlay network topology without ISP boundary

Cost Mode	routingcost				
Cost Metric	numerical				
From/To	1	2	3	4	5
1	0	1	1	2	3
2	1	0	2	3	4
3	1	2	0	1	2
4	2	3	1	0	1
5	3	4	2	1	0

Cost Mode	delay-ow				
Cost Metric	numerical				
From/To	1	2	3	4	5
1	0	2	2	5	6
2	2	0	4	7	8
3	2	4	0	3	4
4	5	7	3	0	1
5	6	8	4	1	0

(a) Routing cost cost matrix

(b) One way packet delay cost matrix

Table 3.3: Example cost map for overlay in Figure 3.5

Without either full administrative control or some multi-ALTO domain orchestration mechanism, a single ALTO server instance is restricted to the information it knows. Being bound by limited topological knowledge, however, does not necessarily mean that valuable inter-layer cooperation is not possible, and will now be subject of discussion. The network map presented previously contains in of itself important information, grouping endpoints into PIDs that represent network borders - one local to the server, and two representing the peering relationships. This is relevant to help clients localize their traffic and would be impossible to derive without insight. Table 3.4 provides an example of cost matrices within a single cost map that consider a limited single

ALTO server domain topology in Figure 3.4. Notice how the administrative domain is within scope of only three of the five network nodes. The ISP only possesses detailed network status information that regards to nodes "1", "3" and "4", which limits the amount of topological information that can be retrieved and shared to ALTO clients - however, it's still very much possible to dictate routing preferences and gaps in knowledge can be filled with probing measurements to be collected and centralized by the ALTO server to acquire historical performance metrics. As can be seen, a generic "routingcost" cost matrix is presented, whose value increases with the associated costs of transferring data through that path, and constructed as the ISP best sees fit - costs within the ISP domain are minimal, whereas paths that originate locally and target "PID2" or "PID5" - both requiring the utilization of peering links - are less preferable, with the former being at least twice more preferable than the latter. A "delay-ow" cost matrix is also provided, specifying one way packet delay in milliseconds, with the ISP applying preceding probing measurements between endpoints and averaging the results. Finally, a "tput" cost matrix can be seen, specifying expected throughput in bytes per second, with the ISP applying probing measurements, topological insight, as well as collected feedback of previous application connections that occurred between endpoints to deduce available bandwidth between target points in practice. Additionally, the inclusion of cost calendar capabilities to the cost matrix enables users to get a chronological view of bandwidth availability at rush hours, with the single value cost being updated to the present time if a decision needs to be made only considering the current time. A similar cost matrix is presented next, that, instead of specifying the throughput costs as dimensional values, does so instead in an ordinal fashion, where the number represents a ranking position in preference, with a lower number representing higher preference. This is an alternative that preserves ranking information without requiring from the the need to concretely specify network status, and instead ordering connections by relative preference, with the option of assigning equal preference to paths that differ in a given order of magnitude that the ISP sees as negligible. In the presented case, locality is correlated with more reliable communications and less operational costs from the ISP's point of view, and a concrete better choice exists - regarding routing cost, delay, and throughput - between the two peering connections. That information can be part of the ALTO system as query-eligible by client applications that can now better optimize their network-related decisions in a mutually beneficial scenario.

Cost Mode	routingcost				
Cost Metric	numerical				
From/To	1	2	3	4	5
1	0	10	1	2	22
2	-	-	-	-	-
3	1	11	0	1	21
4	2	22	1	0	20
5	-	-	-	-	-

(a) Routing cost cost matrix

Cost Mode	delay-ow				
Cost Metric	numerical				
From/To	1	2	3	4	5
1	2	20	1.5	3	42
2	-	-	-	-	-
3	4	24	2	2	39
4	7	27	2	2	36
5	-	-	-	-	-

(b) One way packet delay cost matrix

Cost Mode	tput				
Cost Metric	numerical				
From/To	1	2	3	4	5
1	256000	10000	256000	256000	5000
2	-	-	-	-	-
3	256000	10000	256000	256000	5000
4	256000	10000	256000	256000	5000
5	-	-	-	-	-

(c) TCP throughput cost matrix

Cost Mode	tput				
Cost Metric	ordinal				
From/To	1	2	3	4	5
1	1	2	1	1	3
2	-	-	-	-	-
3	1	2	1	1	3
4	1	2	1	1	3
5	-	-	-	-	-

(d) TCP throughput ranking cost matrix

Cost Mode	tput				
Cost Metric	numerical				
Calendar Start	Tue, 20 Sep 2020 17:00:00 GMT				
Calendar Interval size	7200				
Calendar Interval number	6				
From/To	1	2	3	4	5
1	(1,[1,1,1,2,2,1])	(2,[2,2,2,1,1,2])	(1,[1,1,1,2,2,1])	(1,[1,1,1,2,2,1])	(3,[3,3,3,2,2,1])
2	-	-	-	-	-
3	(1,[1,1,1,2,2,1])	(2,[2,2,2,1,1,2])	(1,[1,1,1,2,2,1])	(2,[2,2,2,1,1,1])	(1,[3,1,1,1,1,1])
4	(1,[1,1,2,2,1,1])	(1,[2,2,1,1,1,1])	(1,[1,1,1,2,2,1])	(2,[1,1,2,2,2,2])	(3,[3,3,3,3,2,2])
5	-	-	-	-	-

(e) TCP throughput cost matrix with calendar values

Table 3.4: Example cost map for the limited ISP domain in 3.5

The network status information of an endpoint property map stores the property information of a given endpoint. The ALTO working group's protocol specification [68] does not directly specify what kind of properties are pondered for this map. Following the same design pattern used for the other specified resources, the endpoint property map will have a set of defined properties with associated semantics, and all other properties can be added with the "priv" prefix to designate private properties outside of the considered domain, and thus all semantics and validation rules don't apply. Much like the other resources, an endpoint can be identified by an ipv4, ipv6, MAC or private overlay address, and the pondered properties are PID value, geographical coordinates, connection type (fiber, *Asymmetric digital subscriber line (ADSL)*, etc.), server footprint information (total RAM, CPU, and storage), and server status information (what portion of the footprint information is currently available, such as free processing power). In practice, a given property could be promoted from a private type to one pondered in the protocol and have a resulting official semantic and validation rules. Table 3.5 display an example endpoint property map, which is used to store status information relating to servers, identified by their ipv4 address, that serve the same content.

Endpoint	CPU %	RAM %	Geographic Coordinates	Connection type	priv:is-mirror
145.132.164.101	22	50	(34.28278,-82.50490)	Fiber	False
245.217.176.67	30	45	(23.24178,-53.51290)	Fiber	True
48.43.96.168	25	30	(55.33218,-12.50490)	Fiber	True
207.20.148.21	10	20	(-23.28121,-22.55530)	Fiber	True
89.140.253.77	5	0	(12.231278,75.70890)	Fiber	True

Table 3.5: Example endpoint property map for server replicas

Finally, as a means to facilitate resource divulgence from servers to clients, there is also included the specification of an *Information Resource Directory (IRD)*, that is also based from the ALTO working group's protocol specification [68]. An IRD can also be thought of as a resource, but instead of sharing network information, it serves as an index of the available resources that a given server provides. Each server must have available for query a single IRD, that lists all the available resources it provides, along with their metadata. Each resource attribute must contain the resource's ID, its HTTP media type and, if applicable, their capabilities, accepted input media types, and resource dependencies. The capabilities identifies, if existing, the cost and property types that are used - being indexed by their unique name, this allows for these to be cross-referenced on further protocol exchanges without need to repeat information. Additionally, the resource's capabilities also serve to indicate what resource functional-

ity extensions are enabled - currently applicable for cost maps only, it serves to signal if the cost map has calendared costs - a protocol extension adapted from the work in progress in [67], that serves to retrieve calendar cost values, or if the multi-cost extension functionality exists - a protocol extension adapted from [87], which lets multiple matrices be requested at once to save on overhead traffic that would otherwise be necessary to request many matrices. Two additions are made to the working group's specification - firstly, a description field, which for each resource attribute gives a brief description of what it is about, as it could facilitate resource selection, since such a description could go into detail about appropriate usage guidelines of that resource and suggested use cases; finally, the resource's ACL, letting a user know beforehand what clearances the given resource has - being a crucial part of the metadata, it seems fitting to go into the IRD, and has the added benefit of giving the user this piece of information without him having to make resource requests just to, by server reaction, figure out what he can and cannot do. A default network map entry must also exist in the IRD, as per the working group's specification, to serve as a guideline for clients that wish to use the most basic of ISP endpoint groupings.

An example IRD is provided in 3.6. A list of available costs and properties is presented, with their descriptive data discussed above, along with the available resources provided by that server, which contains data useful for their server cluster, as well as a broader-purpose endpoint cost map to query for path connection types and facilitate user selection. Finally, a default network map is presented.

Cost ID	Cost Mode	Cost Metric	Description
routing	routingcost	numerical	Default routing preference
routing-rank	routingcost	ordinal	Routing preference by ranking
owd	delay-ow	numerical	Expected one way delay of a single packet. Based on application statistics
tput-theoretical	tput	numerical	Theoretical maximum available TCP throughput. Based on topological knowledge
tput-practical	tput	numerical	Practical expected TCP throughput. Based on application statistics

(a) Available cost types

Property ID	Property type	Description
cpu	CPU	Machine's current CPU load
ram	RAM	Machine's currently occupied RAM
coord	geographic-coordinate	Machine's geographical coordinates
connection	connection-type	Machine interface's connection type
is-mirror	priv:is-mirror	Flag stating if machine is a mirror of original server

(b) Available property types

Resource ID	URI	Media Type	Uses	Accepts	Capabilities	Description
def-nmap	resources/networkmaps/default	alto-networkmap	-	alto-networkmapfilter	-	Default
cluster-costmap	resources/costmaps/cluster	alto-costmap	def-networkmap	alto-costmapfilter	Costs: [routing, routing-rank]	For main data center cluster
cluster-endprop	resources/endpointpropmaps/cluster	alto-endpointprop	-	alto-endpointpropparams	Properties: [cpu, ram, coords]	For main data center cluster
client-endcost	resources/endpointcostmaps/	alto-endpointcost	-	alto-endpointcostparams	Costs: [routing-rank, owd, tput-practical]	For user application guidance

(c) Available resources

Resource ID
def-nmap

(d) Default Network Map

Table 3.6: Example of an ALTO server's IRD

Further formal specification is not made as it has been extensively done in the ALTO protocol [68], and the proposed system complies to it whilst extending upon the design.

3.4 ROLES

3.4.1 ALTO Client

An ALTO resource consumer is materialized in the architecture in the form of an ALTO client, which can be any entity who is able to interface with an ALTO server to query for ALTO resources. Whilst the ALTO working group was initially devised to help increase P2P-related traffic localization via the sharing of network information, it now has an increased scope where an ideal client is any application which generates network traffic and would be able to optimize it with aid from an oracle entity with privileged network information. Thus, an ALTO client is fit to be implemented in P2P applications, and could be embedded in a P2P client itself to help with picking neighbouring and content providing nodes, or on a tracker that would accomplish the same goal on behalf of the querying peer. Likewise, nodes which are unable to optimally select between other nodes, such as CDN edge nodes or content mirrors, could also benefit from oracle guidance, and thus qualify as appropriate ALTO clients.

Figure 3.6 exemplifies how a cooperative P2P application would, acting as an ALTO client, interact with the ALTO server to retrieve relevant network resources to aid their application choice of what candidate peer to consume a service from. Firstly, a network map is retrieved to help group endpoints into groupings, and afterwards a cost map is retrieved filtering only the querying peer as source, candidate peers as destinations, and the routing cost and bandwidth cost matrices. Acting on this information, the peer chooses the candidate that gives a good balance between ISP routing cost and path bandwidth, making a decision that should ideally benefit both them and the ISP that helped provide that information.

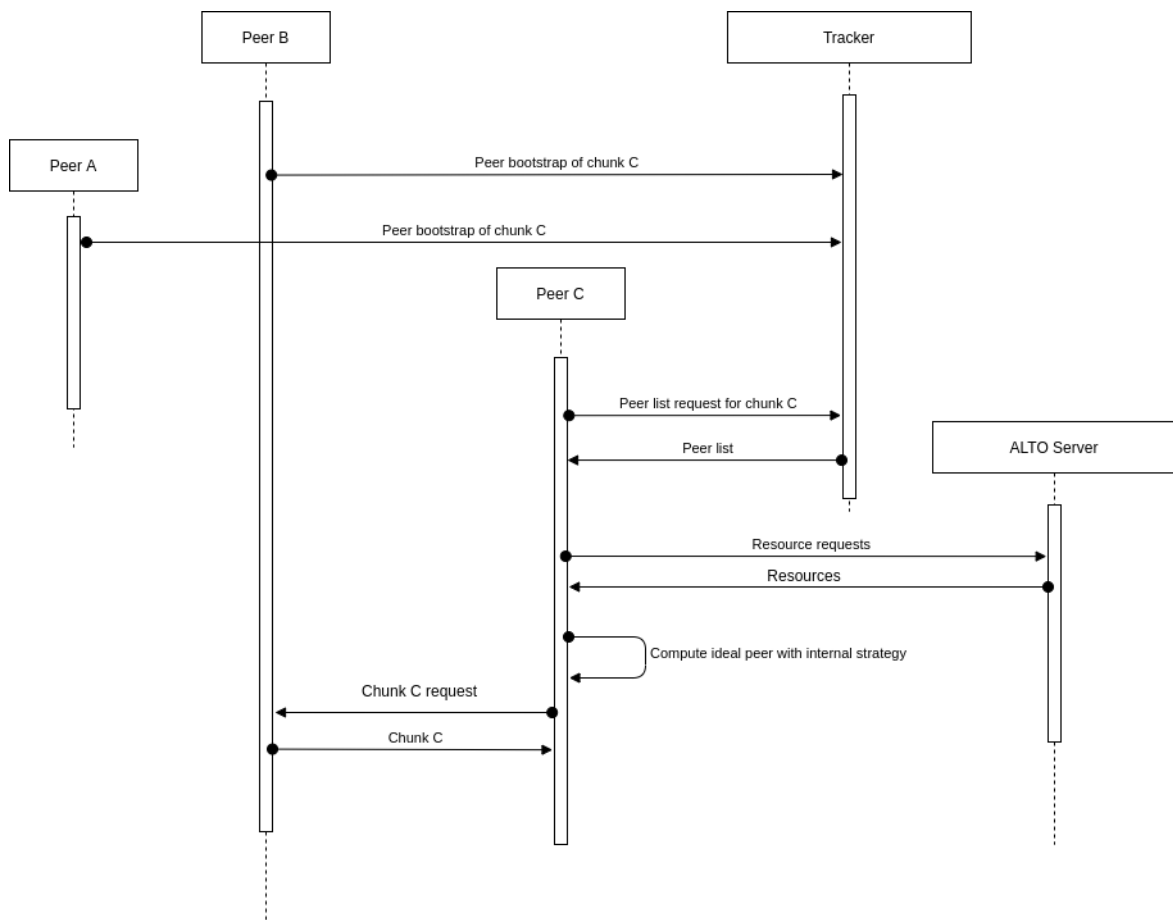


Figure 3.6: High-level communication diagram of a P2P application utilizing ALTO

Figure 3.7 is similar to the previous example, in the sense that it aids a P2P application by resorting to ALTO's guidance, but this time the application-level traffic optimization is made in a way that is transparent to the P2P client. As a choice to purely localize traffic - as this alone can bring plenty of benefits to both layers - and as a means to minimize protocol modification, it is the tracker that acts as an ALTO client. Whenever a request is made by a P2P client to retrieve peers serving a given data chunk, the tracker first consults with the ALTO server and retrieves its network map that groups peers within administrative domains - either inside the providing ISP's domain, thus the local network, and outside administrative domains, grouped by types of peering connections to different autonomous regions. The tracker could use a very simple algorithm to filter out of its candidate pool peers that reside outside of the ISP region where the requesting P2P client resides, if a local alternative exists.

After packaging a reply to the P2P client, the protocol acts normally and traffic could be successfully localized with minimal impact.

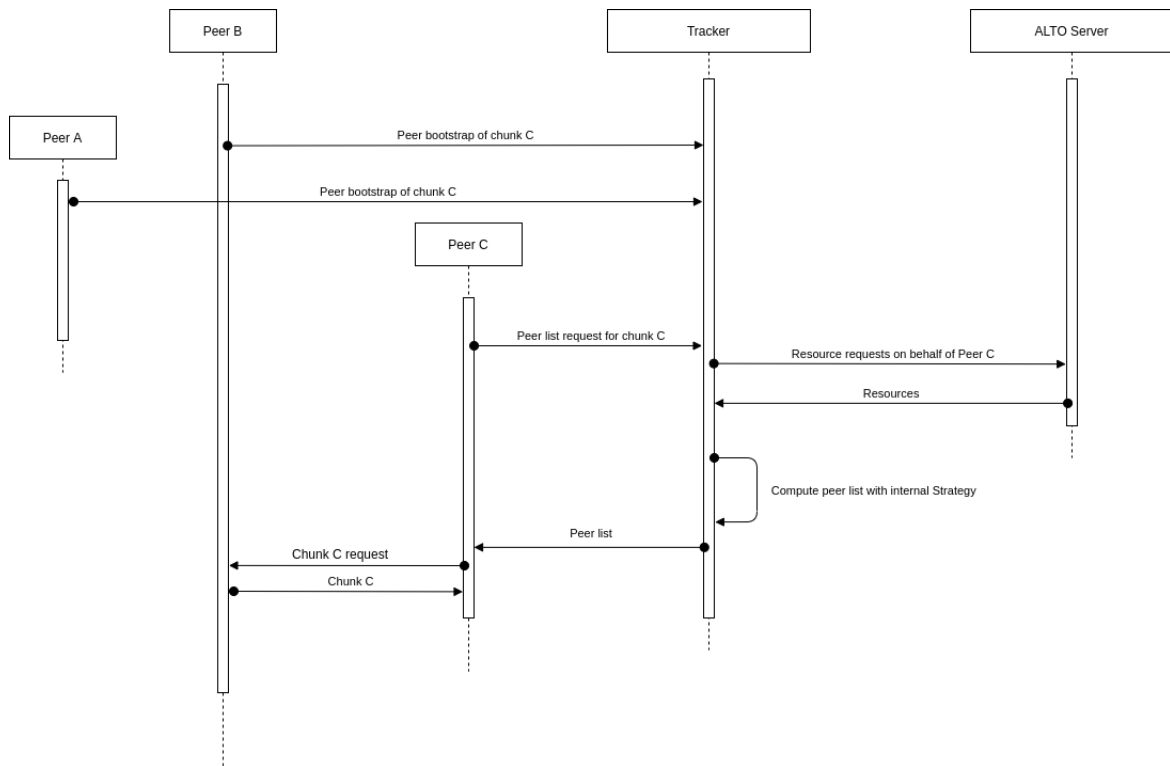


Figure 3.7: High-level communication diagram of a tracker utilizing by proxy

On the same vein, Figure ?? exemplifies how this time a CDN controller would use the system to better help its decision in matching CDN clients to an edge server on their system. To do this, it retrieves a property map to query for server status information, and subsequently retrieves a cost map to query for path information between the CDN client and the candidate edge servers. Having all the relevant server status information, e.g. available processing and storage resources, as well as connection properties, e.g. max possible bandwidth, latency, and packet loss, the CDN controller is in a condition to more optimally redirect his client.

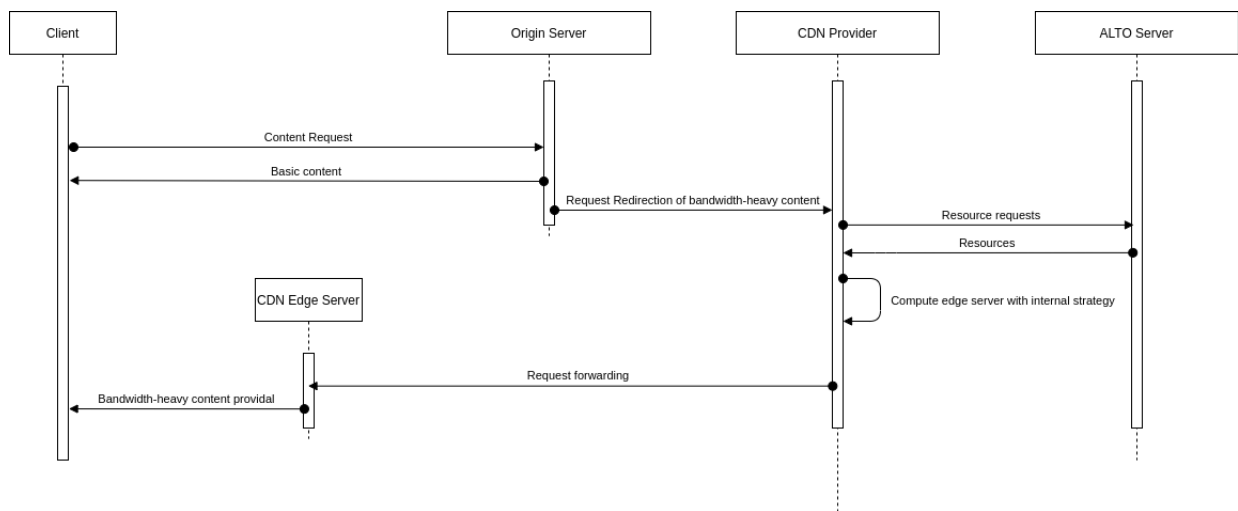


Figure 3.8: High-level communication diagram of a CDN controller utilizing ALTO

3.4.2 ALTO Server

An ALTO resource provider is the ALTO server, an entity that possesses pre-processed and authorized network information in the form of ALTO resources. Its job is to store and manage such resources so they can be provided to querying ALTO clients, with the additional responsibilities of data validation and persistence. Conceptually, the ALTO server is seen as a single entity, but considering the sensible information that could be stored within it and the influence it has on shaping network traffic, it would not be uncommon for an ALTO server to have a knowledge domain correspondent to the ISP that owns it. Physically, though, the resource provider layer could consist of many interlinked ALTO providers with an increased coverage area of network knowledge. Means through which this could occur are further specified in Section 3.4.2.3.

A listing of available HTTP endpoints of the ALTO server interface is available in Table 3.7. All resource types hierarchically descend from a "resources" path, and each unique type of resource exposes his own endpoint, with the methods to add, update, remove, and retrieve with and without filter as arguments. These methods are subjected to the access control mechanisms discussed in Section 3.2, as one should only expect reputable sources to upload and modify data, and only permitted users to query it.

HTTP Verb	Resource	Description
GET	/resources	Retrieve the Information Resource Directory for that server
GET	/resources/networkmaps	Get summary overview of all available network maps
POST	/resources/networkmaps/	Add a new Network Map
GET	/resources/networkmaps/{id}	Get the Network Map with the specified ID
POST	/resources/networkmaps/{id}	Get the Network Map with the specified ID, with the applied filter provided in body
PUT	/resources/networkmaps/{id}	Modify the contents of a Network Map with the specified ID
DELETE	/resources/networkmaps/{id}	Remove a Network Map with the specified ID
GET	/resources/costmaps	Get summary overview of all available cost maps
POST	/resources/costmaps/	Add a new Cost map
GET	/resources/costmaps/{id}	Get the Cost Map with the specified ID
POST	/resources/costmaps/{id}	Get the Cost Map with the specified ID, with the applied filter provided in body
PUT	/resources/costmaps/{id}	Modify the contents of a Cost Map with the specified ID
DELETE	/resources/costmaps/{id}	Remove a Cost Map with the specified ID
GET	/resources/endpointpropmaps	Get summary overview of all available Endpoint Property Maps
POST	/resources/endpointpropmaps/	Add a new Endpoint Property map
GET	/resources/endpointpropmaps/{id}	Get the Endpoint Property Map with the specified ID
POST	/resources/endpointpropmaps/{id}	Get the Endpoint Property Map with the specified ID, with the applied filter provided in body
PUT	/resources/endpointpropmaps/{id}	Modify the contents of an Endpoint Property Map with the specified ID
DELETE	/resources/endpointpropmaps/{id}	Remove an Endpoint Property Map with the specified ID
GET	/resources/endpointcostmaps	Get summary overview of all available Endpoint Cost Maps
POST	/resources/endpointcostmaps/	Add a new Endpoint Cost Map
GET	/resources/endpointcostmaps/{id}	Get the Endpoint Cost Map with the specified ID
POST	/resources/endpointcostmaps/{id}	Get the Endpoint Cost Map with the specified ID, with the applied filter provided in body
PUT	/resources/endpointcostmaps/{id}	Modify the contents of an Endpoint Cost Map with the specified ID
DELETE	/resources/endpointcostmaps/{id}	Remove an Endpoint Cost Map with the specified ID

Table 3.7: ALTO server's available endpoints

3.4.2.1 Resource Filtering

Resource filtering is the task through which a resource consumer can pass a filter object to the resource provider that specifies the parameters that the consumer wishes to retrieve specifically. With it, there's no need to pass more information to the client than he wishes to get, thus minimizing used network bandwidth and client CPU cycles to send and process the resource, respectively. The need for filtering becomes greater at a larger system scale - with an increased number of users that query routinely, and resources that can have a massive amount of entries - specifically those that regard to network endpoints - such a mechanism becomes a necessary optimization. Keeping with the objective of implementing a fully compatible ALTO protocol as specified by the working group, so will the resource filtering specifications be equal to those already specified in the protocol design [68], thus no further specification will be necessary.

For clarification, the ALTO server must maintain an endpoint for retrieving all main specified network status resources via filtering, i.e. the server must let the client retrieve ALTO resources with parametrization that dictates what concrete fields must be delivered. The types of resource filters considered are the following:

- **Network Map filter:** List of of value *PID_List*, that if empty signifies the entire subset of available . All entries of value:

$$(PID, Endpoint_List), PID \in PID_List$$

must be retrieved, and all others must be filtered out.

- **Endpoint Property Map filter:** Pair of list of endpoints and list of properties that must be selected, of value $(Endpoint_List, Property_List)$. All entries of value

$$(Endpoint, Property), Endpoint \in Endpoint_List \wedge Property \in Property_List$$

must be retrieved, and all others must be filtered out.

- **Cost Map filter:** Tuple of list of source , list of destination , list of cost types and list of cost value conditionals of value

$$(SrcPID_LIST, DstPID_LIST, CostType_List, CostValueConditional_List),$$

with $CostType_List > 1$ assuming a multi-cost map extension and the emptiness of any of the lists signifying the entire subset of available values. All entries of value

$$(Src_PID, Dst_PID, Cost_Type, Cost_Value), \\ Src_PID \in SrcPID_List \wedge Dst_PID \in DstPID_List \wedge Cost_Type \in \\ CostType_List \wedge satisfies_atleast_one(Cost_Value, CostValueConditional_List)$$

must be retrieved, and all other must be filtered out.

- **Endpoint Cost Map filter:** Equal to the cost map filter, but considering a list of source and destination endpoint filters instead of .

By analogy, one can consider the ALTO server to act as a remote database with an interface for clients to interact with it, and the filters act as selection statements, such as the "SELECT" method in SQL databases, to retrieve specific parts of the dataset. The filters of cost maps and endpoint cost maps also include a list of premises which themselves are logical operators applied to the candidate cost values. Continuing the analogy, these allow the clients to use "WHERE" statements on the numerical cost values that are retrieved. In summary, the filter functionality could be explained in the examples shown in Table 3.8.

HTTP Verb	Resource	Body	Description
POST	resources/networkmaps/default	<pre>{ "pids": ["PID1", "PID2"] }</pre>	Retrieve an network map with id "default", filtering the entries for "PID1" and "PID2"
POST	resources/endpointpropmaps/default	<pre>{ "endpoints": ["ipv4:10.0.0.1", "ipv4:10.0.0.2"], "properties": ["CPU", "RAM", "connection-type"] }</pre>	Retrieve an endpoint property map with the id "default", filtering the entries for the properties "CPU", "RAM", and "connection-type" and the ipv4 endpoints "10.0.0.1" and "10.0.0.2"
POST	resources/costmaps/default	<pre>{ "cost-type": { "cost-mode": "numerical", "cost-metric": "routingcost" }, "pids": { "srcs": ["PID1"], "dsts": [] } }</pre>	Retrieve the cost map with the id "default", filtering the entries for the cost matrix with cost mode "numerical" and cost metric "routingcost", whose entries have source PID "PID1" and any destination
POST	resources/costmaps/default	<pre>{ "multi-cost-types": [{ "cost-mode": "numerical", "cost-metric": "routingcost" }, { "cost-mode": "ordinal", "cost-metric": "routingcost" }], "pids": { "srcs": ["PID1", "PID2"], "dsts": ["PID3"] } }</pre>	Retrieve the cost map with the id "default" and, assuming a multi-cost protocol extension, retrieve both the "numerical" and "ordinal" variations of the "routingcost" metric, whose entries have source PID "PID1" or "PID2", and destination "PID3"
POST	resources/costmaps/default	<pre>{ "multi-cost-types": [{ "cost-mode": "numerical", "cost-metric": "routingcost" }, { "cost-mode": "numerical", "cost-metric": "delay-ow" }], "calendared": [false, true], "pids": { "srcs": [], "dsts": ["PID3"] } }</pre>	Retrieve the cost map with the id "default" and, assuming a multi-cost and cost calendar protocol extensions, retrieve the numerical variants of the "routingcost" and "delay-ow" metrics, requesting a singular value and a calendar value respectively, of all entries whose destination is "PID3"
POST	resources/costmaps/default	<pre>{ "multi-cost-types": [{ "cost-mode": "numerical", "cost-metric": "delay-ow" }, { "cost-mode": "ordinal", "cost-metric": "routingcost" }], "pids": { "srcs": ["PID1"], "dsts": ["PID2"] }, "or-constraints": { ["[o] ge 0", "[o] le 20", "[1] eq 1"] } }</pre>	Retrieve the cost map with the id "default" and, assuming a multi-cost protocol extension, retrieve the "numerical" mode of the "ow-delay" metric and the "ordinal" mode of the "routingcost" metric, whose entries have source "PID1" and destination "PID2", and whose cost values satisfy for a given source and destination pair that either the "ow-delay" is within 0 and 20 ms or it's the number one preferential "routingcost" value

Table 3.8: Example ALTO queries with the filtering functionality

3.4.2.2 *Server discovery*

ALTO server discovery, by hand of either network information aggregators or resource consumers, must be done leveraging existing DNS technologies. Each server entity maintains a given domain name, and along it is included the need to also maintain domain records in the chosen DNS system to map the domain name to the server's IP address. Much like the choice to utilize HTTP as an application protocol, so do the server discovery mechanisms aim to comply with the ALTO working group's philosophy of leveraging existing proven technologies when possible as a means to facilitate development and minimize errors, with the added benefit of extending functionality with the chosen technologies since it is mature and has plenty of options. With DNS, this gives flexibility of either privately configuring domain name to IP addresses - much like happens in Linux systems with the `"/etc/hosts"` file - or its deployment by leveraging existing authoritative DNS servers. Additionally, by working around the existing technology and its specification, one can easily implement load balancing for performance reasons, or, among others, *DNS over HTTPS (DoH)* for preventing eavesdropping, and ensuring both data integrity and host authenticity [88]. A similar approach is to be taken for network information aggregator server discovery by part of the network state collectors for the same reasons explained above.

3.4.2.3 *Inter-server communication*

A glaring gap in the working group's base ALTO protocol is its single administrative applicability domain. Meaning, an ALTO server is managed by a single administrative entity - likely an ISP - and its knowledge domain is limited by the network topology details that the entity knows, which is a subset of the entirety of the Internet's infrastructure. In the attempt to fix the server's inability to provide network status information outside its domain, this section overviews mechanisms that enable inter-server communications as a means to expand the capabilities of each domain.

Firstly, consider how efforts for full resource synchronization could be taken. These would be similar to data synchronization mechanisms employed by popular databases to ensure consistency across several server replicas, and could increase availability as well as the serviceability of content nearby clients. However, it does not seem to fit this use case - for starters, if all data were to exist redundantly on all servers, that would defeat the purpose of having many administrative domains and thus a single server architecture would suffice; secondly, the architecture is inherently designed to work within a trust domain of selected clients and, because of it, the servers may not even

be comfortable with sharing all of its information within other domains to begin with, limiting replication strategies; thirdly, accounting for the amount of users acting on the ALTO system, better scalability could be achieved with a distributed solution that limits information within set boundaries. Accounting for these reasons, an inter-server synchronization protocol was designed for servers to negotiate information exchange among themselves, as opposed to one that enabled the synchronization of a single monolithic dataset between servers.

It is very important that a multi-domain solution assures that each ISP has full sovereignty over their domain. Simply collecting data from each domain and storing it in an third-party entity from which the original source has no control fails to comply to the sovereignty requirement. Each ISP that participates in a multi-domain knowledge system must, at any time, have control over what the information is and who gets access to it, being able to retroactively change its content and the access control policies, respectively. This solution will then consider that each ALTO server belonging to the multi-domain orchestration mechanism will compute and store it's data locally, and will have an interface open for data querying from other ALTO servers, being then open to selecting and enforcing their own policies as they see fit, in regards to how and when data is calculated, and who gets access to it.

To achieve the stated requirements, a "scope" property will be added in the "meta" field of every ALTO resource. This property, like the name implies, states the scope of the resource - a "local" resource is no different from those in the base protocol, meaning that it gets calculated and distributed within a single administrative ALTO domain; a "global" resource is essentially virtual, in the sense that it is presented as if it were stored in the server, but instead is dynamically retrieved every time through inter-server communication. The decision for this property to be visible in the IRD was made so it is communicated to the client that a given property can be retrieved as a multi-domain effort - and as such is susceptible to a clash of different ISP policies and strategies - or as a locally bound resource that will be less prone to inconsistencies, as only a single domain is responsible for managing it.

To manage the synchronization of globally-scoped ALTO resources, a new entity, the domain synchronizer, was specified. This central server will too be an ALTO server, meaning that it will be used to store and manage ALTO resources, although its use is specialized for inter-server synchronization. The server maintains property maps, where each of them refers to globally and locally scoped resources, and contains the addresses of the servers that independently store that data and make it available for querying. Locally-scoped resource IDs are stored in this server with the addresses of

the server's that host it, so that a given ALTO server, whenever asked for a resource which he does not possess, can contact this synchronization database and appropriately redirect a client to a server that contains that information. In contrast, globally-scoped resource IDs and their owner's addresses are also stored in this server, so that a given server can know who he must query to retrieve the needed information to locally build the resource to be delivered to the client.

Listing ?? shows example data of an endpoint property map for inter-server synchronization. Figure ?? shows the required steps taken by a given ALTO server when a client requests for data - in this case an endpoint property map and a cost map - that is globally scoped.

The required *Application Programming Interface (API)* endpoints that an ALTO server must provide for inter-server sharing of resources is no different from the ones they already must provide for clients. The only difference is that these resources contain ACLs that dictate exclusive action access by authenticated ALTO servers.

3.4.3 Network State Provider

3.4.3.1 Network Information Aggregator

The network information aggregation layer is the layer that enables the translation of raw topological information - such as the physical attributes of network devices and connections - into processed, query-eligible network knowledge. To do so, a very important entity, perhaps the heart of the system as a whole, is the network state collector, which is the supply of network information that is injected, through a network state provisioning protocol, into a network information aggregator. This latter entity is then responsible for providing the ALTO resource provider layer with valid information after the raw topological data has been processed - this includes the calculation of optimal paths, the abstraction of network entities, or the injection of static ISP preferences. This pre-processing stage requires input from an ISP administrator, responsible for acting on the best interest of the ISP from which the raw topological data originates - by interacting with the network information aggregator, the administrator acts on this network information hub to retrieve from the database a history of retrieved network information, and afterwards manipulate this information to create ALTO resources to its liking - this is where data is transformed utilizing the algorithms the administrator deems fitting, and transforms the raw data to be publishing ready, meaning that it contains an acceptable amount of abstraction not to compromise topological privacy.

Finally, the administrator defines important meta data that identifies the resource, and defines the access control list to be enforced by the ALTO server.

3.4.3.2 Network State Collector

Before ALTO resources are provided into the ALTO server by the Network Information Aggregator, the latter needs himself to be provided with raw network status information. The ALTO working group has discussed possible sources of raw topological information, including protocols like IGP, *Border Gateway Protocol (BGP)*, SNMP, or *Network Configuration Protocol (NETCONF)*, or databases like the *Traffic Engineering Database (TED)* or *Label Switched Path Database (LSPD)* [70]. A protocol needs to exist to interface between the entities that collect and provide the raw topological data, and the Network Information Aggregator that processes it and provides it to the ALTO server.

The available endpoints supported by the Network Information Aggregator server are presented in Table 3.9.

HTTP Verb	Resource	Description
POST	/measurements/endpoint	Add a measured endpoint property value
PUT	/measurements/endpoint/{id}	Modify the contents of a measured endpoint property value
DELETE	/measurements/endpoint/{id}	Remove a measured endpoint property value
POST	/measurements/links	Add a measured link value
PUT	/measurements/links/{id}	Modify the contents of a measured link value
DELETE	/measurements/links/{id}	Remove a measured link value
POST	/measurements/group	Add a measured endpoint grouping
PUT	/measurements/group/{id}	Modify the contents a measured endpoint grouping
DELETE	/measurements/group/{id}	Remove a measured endpoint grouping

Table 3.9: Network Information Aggregator’s available endpoints

An illustrative example on how certain Network State Collectors of given network data could use this endpoint to interface with the Network Information Aggregator is presented on Figure 3.9

as an OSPF collector daemon), the time where the measurements were collected, a description, an endpoint or group of endpoints, depending on what the measurement relates to - such as an endpoint property or a cost between two properties or a grouping between N properties, and the measurement itself.]

3.4.3.3 Network Status processing

[Detail how the ISP uses the information gathered by the network state providers and pre-processes it. This includes, for example, calculating shortest path maps utilizing a Dijkstra algorithm, limiting/changing information to maintain security, adding static ISP policies, and attributing access control policies for that resource]

Creating a user-friendly network data constructor would be a good future work

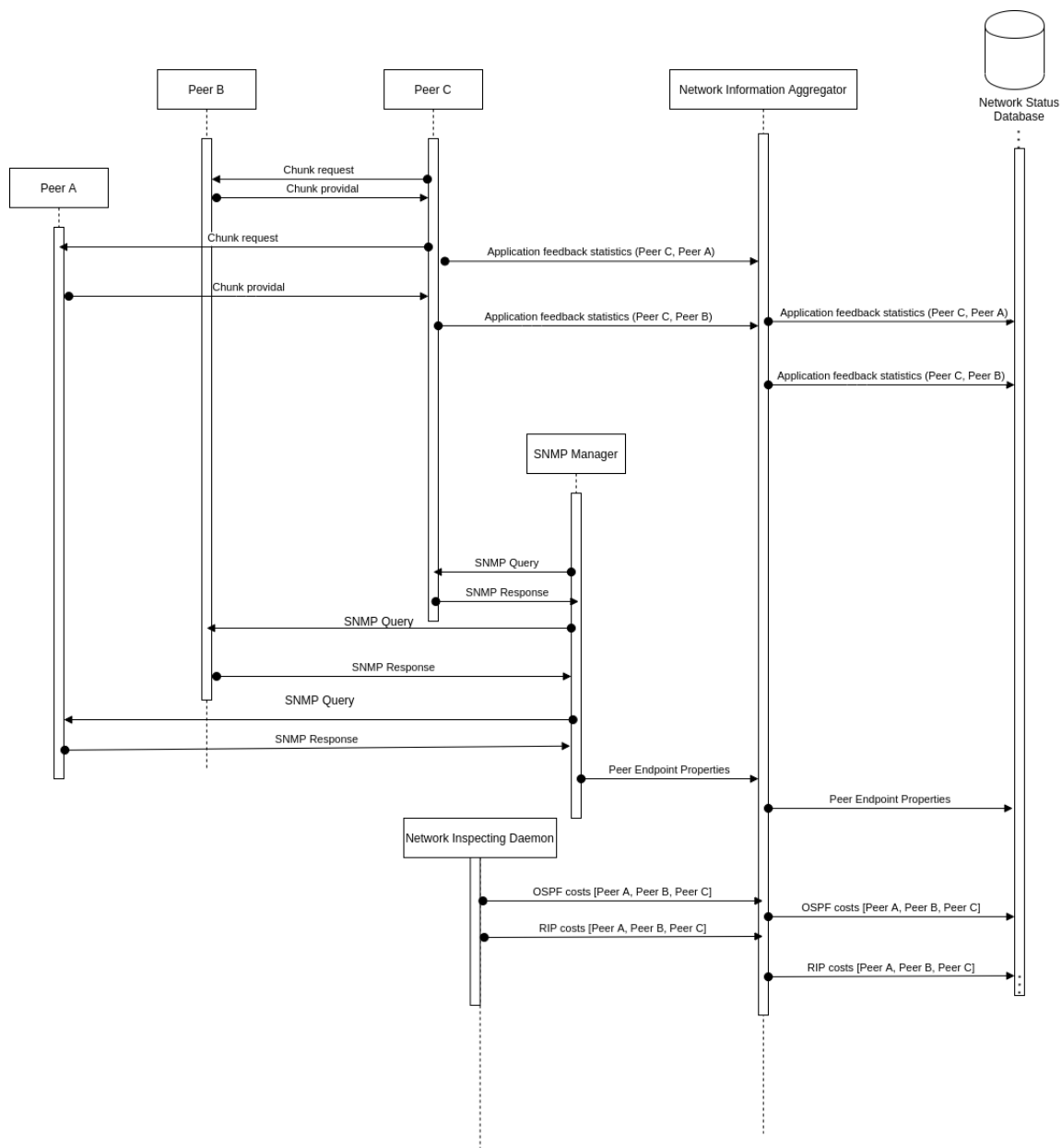


Figure 3.10: Communication diagram of how an ISP administrator pre-processes the gathered network state and uploads it to the ALTO server

4 | IMPLEMENTATION

Following the specification, the aim of this chapter is to overview aspects of the implementation stage of the proposed ALTO system. Firstly, attention is given to the chosen technologies that were leveraged to get the system from its specification stage into a working product - this includes tools and frameworks in the development and deployment phases, and whose choice greatly delimits the system's properties. Secondly, the server is put into spotlight by detailing how it is structured and how it behaves, taking special concern in how object oriented programming was leveraged to maximize modularity and reasoning of the system to facilitate future alterations and extensions. A general special attention is given to the system's security, where an overview is made what concrete steps were taken to nullify the previously detected potential threats that put into question the viability of the system.

4.1 TECHNOLOGIES USED

Starting the implementation stage of every project, attention must be given into what tools are selected to make it come to fruition. These can greatly impact the success of the developed software, and has concrete consequence in its maintenance and future extension.

The specified system architecture is composed by key entities whose interactions among themselves is clearly defined by interfaces. More so, considering the example deployment scenarios, its evident that each entity resides in different topological regions throughout the network, with the ALTO server, clients, and status providers being scattered throughout an ISP domain. Each entity can then be thought of as a self-contained system in of itself who must abide by the proposed interfaces to properly work on the system. A logical conclusion to this is that each entity implementation is independent from the next, needing to only assure a common communication channel that all entities within it can properly understand. This gives great flexibility in the system implementation as a whole, because different tools can be leveraged to differ-

ent entities if needed, and such entities that be worked on independently from the rest without impacting the function of the group.

Regarding the ALTO server, first attention is given to the ALTO resources that must be provided by it. They all share some common properties - such as resource id, ACLs, owner, etc. - and functionality - such as the ability to be read and updated, or their permissions modified. This similarity is further intensified within groups of resource types, more specifically cost maps - consider how the only concrete difference between an endpoint and a PID cost map is the type of entity the costs refer to, which are endpoint and PID addresses, respectively. The sequence of steps that must be taken from the initial point where a client requests a resource up until that resource is provided can be abstracted as the sequential interaction between concrete modules that have a given, self contained, purpose, and communicate with a common interface - an architectural pattern that is a micro version of the one existing in the system, that shares all its properties that were discussed previously. Some of these modules would include client request monitoring, its parsing, its validation, database retrieval, database storage, and serialization. With all this in mind, an object-oriented programming seems like a proper fit for the complexity pertaining to the ALTO server, especially considering how future extensions would be likely, as very much will be in case of the ALTO working group. By working with objects as the base programming entity, many of the highlighted similarities between resources will become easy to be put into evidence, and module encapsulation and interfacing are natural and thus arguably easier to develop and maintain. Importance is also placed in choosing a compiled language that is statically typed - a compiled language favors performance over an interpreted alternative, which seems favourable for the expected scale of the ALTO system, and typed constrictions provide needed structure verifications that aid in the programmer's confidence in its reliability, and often helps reduce mistakes. The language choice finalizes with Java [89], which matches the given requirements and is the one with most prior personal development experience, coupled with its maturity and wide access to libraries and frameworks.

The Java Spring framework [90] is a main component of the developed server software - its inversion of control paradigm allows dependencies to be managed through configuration files and annotations to identify objects to be scanned by the framework, providing a means of development that is quite flexible, reduces boilerplate code, and favors loose coupling between objects, facilitating the independent development of implementations that can themselves be changed in the future with little consequence to the entire system, given that they implement the stipulated interface.

Adding to this, the Spring framework provides modules that are very much needed for the server implementation, namely for *Model-View-Controller (MVC)* architectures, HTTP-based *Representational state transfer (REST)* APIs, authentication and authorization, data access, and both unit and integration testing, to name a few. Given that other frameworks could provide similar functionality, Spring was favored due to its development team's focus on performance and flexibility, and because the framework has continuous development support and an increased community popularity, all of this raising the odds that this framework becomes maintained throughout the future, in comparison to some that have since lost support.

Finally, focus was given for database-related decisions. Since resource manipulation through the REST APIs revolve around *JavaScript Object Notation (JSON)* manipulation, a preference was made for a database that allows the resources to be saved in a similar fashion, thereby reducing a layer of abstraction that makes reasoning on the software easier. Considering the scale at which an ALTO server might be subjected to requests, a database that favors performance seems like a better choice, in particular one that easily allows for horizontal scaling. A last consideration is made for a database that is schema-less - for the network information aggregation server, a myriad of types of data can be added by state providers to account for the huge variety of protocols and standards that retrieve network information, and more can exist in the future that weren't initially pondered by the ISP administrator, as more network protocols and data properties become relevant to attain, so a flexibility in what can be stored seems more appropriate; for the ALTO server, as the ALTO protocol itself has been subject to continuous change that still aim for legacy support, it can be argued that future changes will be common, and doing so without increased server downtime - something that would be required by schema-driven databases - seems favorable considering the scale and importance of an ALTO service. The existence of private properties whose scope is outside of the protocol and can be freely defined the user, while not impossible to implement with a schema, seems to lend itself more naturally to a schema-less design. With all these considerations in mind, a database that abides to these constrictions and has a healthy developer support and user adoption is MongoDB [91], which was chosen.

As per security, the same philosophy of the ALTO working group will be used and pre-existing, mature technologies will be leveraged. *Hypertext Transfer Protocol Secure (HTTPS)* will be used over base HTTP as a communication protocol between entities as a means to resolve many of the identified threats - by using certificates with HTTPS, the server entity authentication can be assured, and through the usage

of *Transport Layer Security (TLS)* as a cryptographic security tool, both confidentiality and integrity are secured via data encryption and digest calculation, respectively. As per client authentication, a deliberate choice was made to use HTTP basic [92], as opposed to HTTP digest [93] that is demanded by the base ALTO working group's specification. With basic authentication, user and password fields are sent in encoded - not encrypted - fashion, and such information is firstly validated by the server that, if the attempt is successful, proceeds with its normal operation. Due to the lack of encryption, this method of client authentication will be complemented with the usage of HTTPS to provide a secure communication channel that would otherwise be open to credential exposure and tampering. The working group's suggested digest method hashes the credentials by using a nonce - a number to be used only once - that was provided by the server, thus also protecting against data exposure and tampering. However, since HTTPS will be leveraged, there's no need to have an authentication system that does much more beyond of what HTTP basic does, therefore facilitating client applications and giving the server flexibility on how they wish to store user credentials - the basic authentication allows one to store the hash of passwords alone, whereas digest requires the storage of the hash value of "username:password:realm". These and other comparisons between these authentication methods are available at [?] and further helped in the decision.

4.2 SERVER ARCHITECTURE

[emphasis on single responsibility and behavior reutilization, as is expected on object oriented design]

The macro-level architectural diagram specified that the server's role is to serve incoming requests by clients and providers, and to interface with a database to persist resource storage. The server will implement a REST interface leveraging the HTTP as this pair is widely accepted and ubiquitous on the Internet, but also due to the fact that its resource-oriented interface standards fit nicely into the specified interface for ALTO server interactions, which too revolve around resource manipulation, and by adhering to proper REST designs good scalability can be achieved due to its stateless nature and potential for resource caching. The choice of HTTP as an application protocol fits nicely into a philosophy of leveraging existing and well proven protocols and technologies to increase the project's success, and indeed so was done to integrate authentication and encryption mechanisms.

To accomplish this interface implementation, the internal server architecture will be, at a macro level, as shown was in Figure [?]. This three-layered architecture consists firstly by a controller layer that intercepts communication requests, which after parsed and validated are redirected to the business layer, which in turn employs business logic to help satisfy the controller's requests, which may require a subsequent layer descent into the data layer via database queries.

Figure [?] displays a class diagram focusing on controller classes that deal with resources that are not susceptible to version control - this includes every resource except the network map. As can be seen, all concrete controllers - such as an endpoint property map controller - are extensions to a generic controller class that is parametrized by its *Data Transfer Object (DTO)* , filter DTO, and service instances. This design choice was made because all controller logic that regards to resources without version control are the same, and by creating generic classes with type parametrization code reutilization is increased. The parametrization required by the controller is required to pass a concrete instance of the unversioned resource service generic class, which in of itself requires parametrization in resource DTO and resource filter DTO. By reflecting on common controller and service behaviour between all version control lacking resources, the conclusion was that working around generic classes maximizes reutilization, facilitates reasoning and decreases potential error. To help better visualize the result, refer to how the generic controller is implemented in Listing ?? . To retrieve a resource, simply call the service class with or without the proper filter, depending on which method was triggered, by calling the appropriate methods that must implement the resource service interface. For example, an endpoint property map controller implementation simply extends the generic controller by providing the concrete DTO, filter DTO, and service implementations, as seen in Listing 4.2.

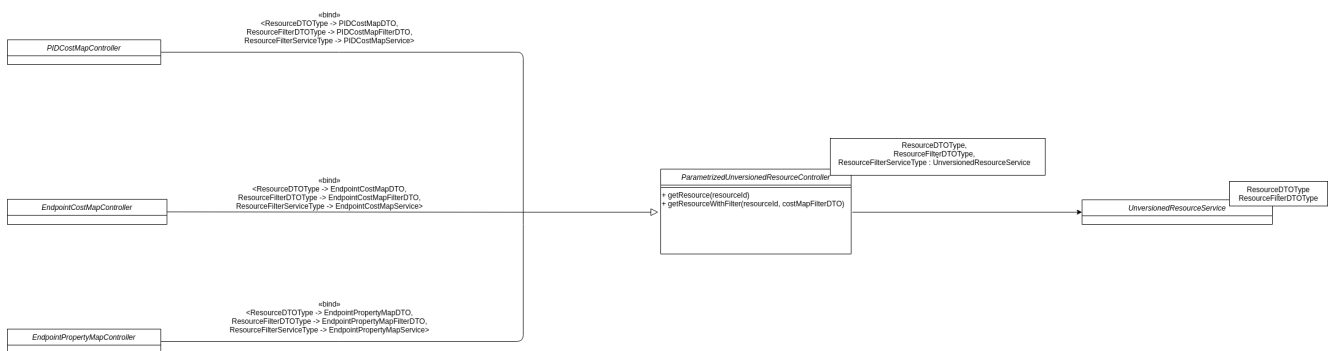


Figure 4.1: Controller layer class architecture

Listing 4.1: Parametrized Controller class for unversioned resources

```
public class ParametrizedUnversionedResourceController
    <ResourceDTOType,
        ResourceFilterDTOType,
        ResourceServiceType extends ALTOUnversionedResourceService
            <ResourceDTOType,
                ResourceFilterDTOType>
    > {

    private final ResourceServiceType resourceService;

    @Autowired
    public ParametrizedUnversionedResourceController(ResourceServiceType resourceService) {
        this.resourceService = resourceService;
    }

    @PreAuthorize("@ResourceAuthorizationService.hasPermission(authentication, #resourceId, T(com.example.
        restservice.dto.security.PermissionDTO).READ)")
    @RequestMapping(method = RequestMethod.GET, value = "{id}")
    public ResourceDTOType getResource(@PathVariable(value = "id") String resourceId) {
        return resourceService.getResource(resourceId);
    }

    @PreAuthorize("@ResourceAuthorizationService.hasPermission(authentication, #resourceId, T(com.example.
        restservice.dto.security.PermissionDTO).READ)")
    @RequestMapping(method = RequestMethod.POST, value = "{id}")
    public ResourceDTOType getCostMapWithFilter(@PathVariable(value = "id") String resourceId,
        @Valid @RequestBody ResourceFilterDTOType costMapFilterDTO) {
        return resourceService.getResource(resourceId, costMapFilterDTO);
    }
}
```

Listing 4.2: Concrete controller extending from a parametrized one

```
@RestController
@RequestMapping("endpointprops")
public class EndpointPropertyMapController
    extends ParametrizedUnversionedResourceController<EndpointPropertyMapDTO, EndpointPropertyMapFilterDTO,
        EndpointPropertyMapService> {

    @Autowired
    public EndpointPropertyMapController(EndpointPropertyMapService resourceService) {
        super(resourceService);
    }
}
```

The same reasoning was used to implement the network map controller, but since this is the only resource that accepts versioning, the correspondent generic controller behavior does not contain similar behavior to the one above, and thus another was created, as seen in Listing ???. The service layer implementation must now let the controller retrieve either a specific version of a resource, or the most recent one if no version is specified by the client.

Listing 4.3: Parametrized controller class for versioned resources

```
public class ParametrizedVersionedResourceController
    <ResourceDTOType,
    ResourceFilterDTOType,
    ResourceServiceType extends ALTOVersionedResourceService
        <ResourceDTOType,
        ResourceFilterDTOType>
    > {

    private final ResourceServiceType resourceService;

    @Autowired
    public ParametrizedVersionedResourceController(ResourceServiceType resourceService) {
        this.resourceService = resourceService;
    }

    @RequestMapping(method = RequestMethod.GET, value = "{id}")
    @PreAuthorize("@ResourceAuthorizationService.hasPermission(authentication, #resourceId, T(com.example.
        restservice.dto.security.PermissionDTO).READ)")
    public ResourceDTOType getVersionedResource(@PathVariable(value = "id") String resourceId,
        @RequestParam(value = "version", required = false) String
            resourceVersion) {

        return resourceVersion != null
            ? resourceService.getResourceVersion(resourceId, resourceVersion)
            : resourceService.getLatestResourceVersion(resourceId);
    }

    @PreAuthorize("@ResourceAuthorizationService.hasPermission(authentication, #resourceId, T(com.example.
        restservice.dto.security.PermissionDTO).READ)")
    @RequestMapping(method = RequestMethod.POST, value = "{id}")
    public ResourceDTOType getVersionedResourceWithFilter(@PathVariable(value = "id") String resourceId,
        @RequestParam(value = "version", required = false) String
            resourceVersion,
        @Valid @RequestBody ResourceFilterDTOType resourceFilter)
    {

        return resourceVersion != null
            ? resourceService.getResourceVersion(resourceId, resourceVersion, resourceFilter)
            : resourceService.getLatestResourceVersion(resourceId, resourceFilter);
    }
}
```

The purpose of some pieces of the shown code isn't immediately obvious, and will be now further explained. Starting with input validation, the usage of the "@Valid" annotation on an endpoint controller parameter signifies that the framework must initialize a validator class that should examine and validate that input against the defined rules. The rules are defined on the DTO class that is to be passed by the client into the server, and validation annotations from the framework are leveraged to define attribute-specific rules. ??, for example, includes the "@JsonProperty" annotation that defines optional and obligatory fields in the de-serialization step, and whenever needed more types of attribute validation annotations were used, including non-blank strings, non negative integers, and values within a certain non continuous set of possibilities. For class-wide validation, custom validator classes were created and afterwards annotated into the class in question - for the class in Listing ??, which defines the filter DTO that is passed by the client whenever cost map filtering is selected, a

custom validator class was created and annotated with "@ValidCostParametrization". This class, seen in ??, implements the class-wide restrictions imposed in the ALTO protocol regarding cost map filters, which dictate that a user can only select either a single or multi-cost map request, and whenever calendarization is requested, the total count of cost types requested must match the number of flags that dictate if a given cost type must provide its calendar form. Assuring both attribute and class-wide rule definition and enforcement, the system can validate all user input to maintain system correctness.

Another element from the shown controller implementation is the "@PreAuthorize" annotation. These, and similar others, are part of the Spring security module and serve as constriction setters that, if not successful, do not let the annotated method be executed. This is the basis for access control in the server system - as 4.3 shows, retrieving a resource using the GET method requires that a given constraint is verified. This constraint verification process is handled by a specially created authorization service that receives authentication information, the resource id in question, and what action is being requested - in this case being "READ". This is how all access control restrictions upon ALTO resources is implemented - with uploaded resources containing an ACL mapping user roles to allowed actions, all controller access must first verify that the given authorized user is authorized to perform that action. The authorization service, seen on ??, exposes a "hasPermission" method in its interface and is tasked with confirming if the following action is allowed considering the system's constrictions and the upload entity's defined access control rules. Simply put, for a service to validate a request it must validate that the following are all true: a) the user is authenticated; b) the user has general system access to the system with that action and c) the user has concrete access to that resource with that action. By creating an additional access layer to the system that is separate from the resource's access list, the server can apply both a per-role and per-resource access control, allowing more control for server administrators outside of what the resource dictate - for example, completely blacklisting a given role from a system even if a resource upload entity allows him with his ACL.

As per the service layer, a similar philosophy to the controller was taken - i.e., to expose behavior similarities and utilize generic types to inject dependency implementations. Listing ?? displays the class diagram of services that deal with unversioned resources, and Listing ?? shows the versioned side.


img/service-unversioned-architecture.png

Figure 4.2: Unversioned service layer class architecture

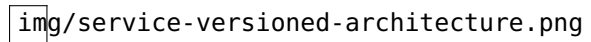


Figure 4.3: Versioned service layer class architecture

Much like the controller architecture, by isolating common patterns one can minimize code repetition and promote function modularity, and this results in faster development in less errors that could result in redundant logic being applied. As can be seen, service classes leverage resource repositories and mapper classes that translate an entity into a transfer object, which is a helpful design decision that decouples class implementation from representation, allowing the system to comply to the defined protocol without the restriction of storing and handling it the same way within the system. A versioned service is quite similar, but exposes an interface that lets the caller retrieve resources considering their version tag, or simply retrieve the most recent version.

As an example, Listing ?? shows the cost map service implementation, that simply extends the parametrized class and injects the implementations specific to how a cost map service must behave - i.e., how it stores and how it translates resources. It then adds only methods specific to their service implementation that aren't shared among others - in this case, how to retrieve a resource with a given cost map filter, which requires different actions depending on whether or not the filter specifies a single or multi cost request.

The mapper classes, mentioned above, are tasked with mapping between two different representations of a class. Specifically, mapper classes are used in the server to map between protocol representations of a resource or a filter, into representations to be used internally. Like previously mentioned, decoupling protocol and internal class representations makes the resources easier to store, as they can be translated into a form more ideal for MongoDB storage - in some cases making some queries quite impossible to achieve otherwise - and easier to handle, since protocol representation of data is better for data transmission and user readability, but not as much for querying and algorithmically processing. For example, the protocol representation of calendarized cost maps separates the cost information, cost values and calendar information into three separate lists, and their matching must be made by equal index access. Working internally with a data structure that is optimized for simpler and quicker data handling and querying can optimize application performance and reduce code complexity, being a good compromise for the added mapping layer that is consequentially required. As an example, Listing 4.4 display how a calendar cost map is represented in the ALTO protocol versus that same content as a storage-ready en-

tity. Whilst still storing the same information, changing how it's laid out could facilitate human reading and client parsing for a display-oriented usage in the protocol representation, and facilitate traversal and general database query for a storage-oriented usage.

<pre> /calendar/costmap/filtered HTTP/1.1 200 OK Content-Length: 1043 Content-Type: application/alto-costmap+json { "cost-type" : { "cost-mode" : "numerical", "cost-metric" : "routingcost", }, "calendar-responses-attributes" : [{ "calendar-start-time" : "Tue, 1 Jul 2019 13:00:00 GMT", "time-interval-size" : 7200, "number-of-intervals" : 12 },], "cost-map" : { "PID1" : { "PID1" : [1, 12, 14, 18, 14, 14, 14, 18, 19, 20, 11, 12], "PID2" : [13, 4, 15, 16, 17, 18, 19, 20, 11, 12, 13, 14], "PID3" : [20, 20, 18, 14, 12, 12, 14, 14, 12, 12, 14, 16] } } } </pre>	<pre> { "resourceId" : "filtered", "mappingEntities": [{ "costMode": "numerical", "costMetric": "routingcost", "calendarAttributesEntity": { "startTime": "Tue, 1 Jul 2019 13:00:00 GMT", "intervalSize": 7200, "intervalNumber": 12, "iterations": 1 }, }, "fromSrcCostEntities": [{ "srcNode": "PID1", "dstCostEntities": [{ "dstNode": "PID1", "calendarCostValues": [1, 12, 14, 18, 14, 14, 14, 18, 19, 20, 11, 12] }, { "dstNode": "PID2", "calendarCostValues": [13, 4, 15, 16, 17, 18, 19, 20, 11, 12, 13, 14] }, { "dstNode": "PID3", "calendarCostValues": [20, 20, 18, 14, 12, 12, 14, 14, 12, 12, 14, 16] }] }] } } </pre>
---	--

Listing 4.4: Example calendar cost map in protocol and database representation

Data access is materialized in the form of repository classes, responsible for providing an interface for service classes that let them retrieve entity classes from the database, doing so by generating the required queries to the Mongo database. Again, for behaviour similarity exposure, the repositories inherit from a base versioned and unversioned repository that compiles the queries needed to retrieve resources, requiring the inheriting repositories to specify additional implementation-specific queries needed to extend upon the functionality. For this to be possible, entity representation on the database must too be similar, so the base queries, when applied to either a

network map or an endpoint property map, which are two unversioned example resources, can work the same. Listing 4.5 shows an example database representation of a cost map and an endpoint, respectively. Notice how common properties are indexed in the same way, and only differ on implementation-specific details. The "resourceId" attribute can be queried and the "mappingEntities" attribute can be loaded by the high-level query without needing to know what concrete resource entity is being treated, and further implementation-specific querying is made by the inheriting repositories.

```
{
  "resourceId" : "my-default-cost-map",
  "mappingEntities": [
    {
      "addressType" : "IPv4",
      "addressValue" : "192.0.2.34",
      "endpointPropertyEntities" : [
        {
          "propertyType" : "my-default-network-
            map.pid",
          "propertyValue" : "PID1"
        },
        {
          "propertyType" : "geolocation",
          "propertyValue" : "123132;33231"
        }
      ]
    }
  ]
}
```

```
{
  "resourceId" : "filtered",
  "mappingEntities": [
    {
      "costMode": "numerical",
      "costMetric": "routingcost",
      "calendarAttributesEntity": {
        "startTime": "Tue, 1 Jul 2019 13:00:00 GMT",
        "intervalSize": 7200,
        "intervalNumber": 12,
        "iterations": 1
      },
      "fromSrcCostEntities": [
        {
          "srcNode": "PID1",
          "dstCostEntities": [
            {
              "dstNode": "PID1",
              "calendaredCostValues": [1, 12, 14, 18, 14, 14, 14,
                18, 19, 20, 11, 12]
            },
            {
              "dstNode": "PID2",
              "calendaredCostValues": [13, 4, 15, 16, 17, 18, 19,
                20, 11, 12, 13, 14]
            },
            {
              "dstNode": "PID3",
              "calendaredCostValues": [20, 20, 18, 14, 12, 12, 14,
                14, 12, 12, 14, 16]
            }
          ]
        }
      ]
    }
  ]
}
```

Listing 4.5: Similar structure in a costmap and endpoint property map storage entities

Listing ?? shows a portion of the network map repository implementation. To retrieve a specific version of a resource, build query methods are retrieved from the versioned repository class, and network map specific processing is added to apply network map projections - which in this case means retrieving only the specified PIDs.

Listing 4.6: Network map repository

```

@Repository
public class NetworkMapMongoRepository extends VersionedResourceMongoRepository<NetworkMapEntity>
    implements NetworkMapRepository {

    public NetworkMapMongoRepository(MongoTemplate mongoTemplate) {
        super(NetworkMapEntity.class, mongoTemplate);
    }

    private Optional<List<AggregationOperation>> givenSingleVersionOfResourceBuildProjectionAggregationOperations(
        NetworkMapProjection networkMapProjection) {
        if (networkMapProjection.getSrcPIDs().isPresent()) {
            return Optional.of(
                Arrays.asList(
                    unwind("mappingEntity.addressAggregationEntities"),
                    match(Criteria.where("mappingEntity.addressAggregationEntities.pid").in(
                        networkMapProjection.getSrcPIDs().get())),
                    group()
                        .first("resourceId").as("resourceId")
                        .first("mappingEntity.versionTag").as("versionTag")
                        .push("mappingEntity.addressAggregationEntities").as("addressAggregationEntities"),
                    project("resourceId")
                        .and("versionTag").as("mappingEntity.versionTag")
                        .and("addressAggregationEntities").as("mappingEntity.addressAggregationEntities")
                )
            );
        } else {
            return Optional.empty();
        }
    }

    private List<AggregationOperation> buildFindResourceAggregationOperations(String resourceId, String versionTag,
        NetworkMapProjection projection) {
        List<AggregationOperation> aggregationOperations = new ArrayList<>();

        if (versionTag != null) {
            aggregationOperations.addAll(buildGetVersionOfResourceAggregationOperations(resourceId, versionTag));
        } else {
            aggregationOperations.addAll(buildGetLatestVersionOfResourceAggregationOperations(resourceId));
        }

        givenSingleVersionOfResourceBuildProjectionAggregationOperations(projection).ifPresent(aggregationOperations
            ::addAll);

        aggregationOperations.add(getWrapVersionInsideArrayOperation());

        return aggregationOperations;
    }

    private List<AggregationOperation> buildFindResourceAggregationOperations(String resourceId,
        NetworkMapProjection projection) {
        return buildFindResourceAggregationOperations(resourceId, null, projection);
    }

    @Override
    public Optional<NetworkMapEntity> findOneFilterByVersion(String resourceId, String resourceVersion,
        NetworkMapProjection networkMapProjection) {
        List<AggregationOperation> aggregationOperations = buildFindResourceAggregationOperations(resourceId,
            resourceVersion, networkMapProjection);

        TypedAggregation<NetworkMapEntity> aggregation = new Aggregation(NetworkMapEntity.class,
            aggregationOperations);

        NetworkMapEntity networkMapEntity = findResourceViaAggregation(aggregation);

        return Optional.ofNullable(networkMapEntity);
    }

    @Override
    public Optional<NetworkMapEntity> findOneFilterByLatestVersion(String resourceId, NetworkMapProjection
        networkMapProjection) {
        List<AggregationOperation> aggregationOperations = buildFindResourceAggregationOperations(resourceId,
            networkMapProjection);

        TypedAggregation<NetworkMapEntity> aggregation = new Aggregation(NetworkMapEntity.class,
            aggregationOperations);

        NetworkMapEntity networkMapEntity = findResourceViaAggregation(aggregation);

        return Optional.ofNullable(networkMapEntity);
    }
}

```

A good API must be aware of possible errors that may occur, and communicate these to the client in a way that is easy to understand. To achieve this, a global exception handling mechanism was used with the aid of the "@RestControllerAdvice" annotation, that is tagged to a class that will consequently catch and process controller-thrown exceptions, which themselves could've been propagated from the service, mapper, or repository layers. It then centralizes the error handling aspect that is then focused in building the correct error packets that contain the appropriate HTTP code and a message with helpful details. These message details are extracted directly from the thrown exception, but it is important to assume that these are to be exposed to the clients, and thus should not contain heavy implementation details for developers, with logging instead taking that role. Listing 4.7 shows the main exception handler used.

Listing 4.7: Main exception handling class

```
@RestControllerAdvice
public class MainExceptionHandler {

    @ExceptionHandler({ Exception.class })
    @ResponseStatus (INTERNAL_SERVER_ERROR)
    public ErrorMessageDTO handleAllOtherExceptions (Exception ex) {
        ex.printStackTrace();
        return new ErrorMessageDTO (INTERNAL_SERVER_ERROR.value(), ex.getMessage());
    }

    @ExceptionHandler (AccessDeniedException.class)
    @ResponseStatus (FORBIDDEN)
    @ResponseBody
    public ErrorMessageDTO PermissionError (AccessDeniedException ex) {
        return new ErrorMessageDTO (FORBIDDEN.value(), ex.getMessage());
    }

    @ExceptionHandler (MethodArgumentNotValidException.class)
    @ResponseStatus (BAD_REQUEST)
    @ResponseBody
    public ErrorMessageDTO validationError (MethodArgumentNotValidException ex) {
        BindingResult result = ex.getBindingResult();
        return new ErrorMessageDTO (BAD_REQUEST.value(), result.getAllErrors().get(0).getDefaultMessage());
    }

    @ExceptionHandler ({ NotFoundException.class })
    @ResponseStatus (NOT_FOUND)
    public ErrorMessageDTO handleResourceNotFoundException (Exception ex) {
        ex.printStackTrace();
        return new ErrorMessageDTO (NOT_FOUND.value(), ex.getMessage());
    }

    @ExceptionHandler ({ MultipleInformationResourceDirectoriesException.class })
    @ResponseStatus (INTERNAL_SERVER_ERROR)
    public ErrorMessageDTO handleMultipleInformationResourceDirectoriesException (Exception ex) {
        return new ErrorMessageDTO (INTERNAL_SERVER_ERROR.value(), ex.getMessage());
    }
}
```

4.3 NETWORK INFORMATION AGGREGATOR

The Network Information Aggregator behaves very similarly to the ALTO server, in the sense that it is an HTTP REST server that exposes an interface for the addition, modification, and deletion of resources, with the actions being pre-validated with an authentication and access control mechanisms. The concrete differences come from the fact that whereas the ALTO server's managed assets are resources, the ones managed by the Network information aggregator are network status measurements. Because of this, most of the implementation decisions remained the same and no technologies were added or removed from those chosen for the ALTO server's implementation.

[How the network information aggregator server was implemented to receive network measurements - interface, validation, security, etc. Will be similar to the ALTO server with the difference that it handles measurement collection instead of ALTO resources collection]

4.4 NETWORK STATE PROVIDERS

[Overviews the implementation of some of the network state providers developed - one listens for OSPF messages, another reads csv files, etc, and all upload to the interface implemented by the network information aggregator]

4.5 SERVER DISCOVERY

[Leveraging DNS like the working group suggests]

5 | EXPERIMENTS

The purpose of this chapter is to overview the experiments phase of the project, that contains the work done to deploy and measure how the system performs in a simulated scenario. Whereas the developed unit tests in the implementation stage aim to verify the correct functionality of separate units of code pertaining to the system, the execution of the entire system as a whole to serve a set of hypothetical use cases can help achieve a better grasp on how correct system functionality between all the tested units. Adjacent to the goal of testing the system in deployed scenarios, the experiments phase also aims to embed in the simulated environment a list of application scenarios that could leverage the ALTO system to its advantage, and subsequently observe and measure if and how the ALTO server can help the client with its network resources that guide the client in taking application decisions that aim for a win-win scenario between the overlay and underlay. As comparison, other known application-network interaction strategies will also be observed and their results measured as a means to compare their impact in comparison to one that utilizes the implemented system. Findings on existing application-layer traffic optimization interactions and the proposal of the ALTO protocol made on section 2, together with the specified system extensions on ?? leads one to believe that a theoretical mutually beneficial scenario exists in an ALTO approach that cannot exist with more asymmetrical means of interaction. This chapter, however, puts those theoretical scenarios into a practical environment that could be replicated by those reading this work, and exposing the created scenarios and collected data can corroborate the theoretical conclusions, as well as leaving an opportunity for future discussion on how the system behaved, including its performance, its success in aiding clients, other existing client options that could be a better route, system shortcomings, etc. This discussion benefits the ALTO project and can give more maturity to the system as it was put through a simulated deployment against other common strategies.

The first section displays the chosen technologies for tasks pertaining to the experiments. The next section focuses on the required steps taken to setup the testing environment. This includes the design and deployment of a network topology in a simulation, the creation of mock applications to serve as clients for the system, and

the design and deployment of application and network status measurement tools. The following section individually overviews the devised scenarios to test in the simulation, and with it experiment specifics such as the initial problem, what strategies will be tested to solve it, how many runs will be made per strategy, and what metrics will be measured. Finished the experiments, the following section will display the obtained results that were collected in the simulated environment, and the section after that will discuss these results and how they fare with the theoretical findings.

5.1 TECHNOLOGIES USED

The *Common Open Research Emulator (CORE)* [?] was used as a network simulator and represents the backbone of the experiments as a whole as it will serve as the background for the running scenarios. This tool allows for the creation and emulation of network environments, and with it are included the abilities to construct network topologies and manipulate properties of the member nodes, which can include network routers, switches, and host machines, that will all be used for the designed experiment scenarios. Additionally, link connection properties can themselves be customized, as parameters like max bandwidth, packet loss percentage, or packet delay can be meticulously customized, and in fact will be in the upcoming scenarios as a means to simulate a given circumstance that may occur in a realistic environment, such as link inefficiency that results from peak traffic hours. Another property that was of great importance for its selection on this work is that, on top of the virtual network environment, arbitrary code can be run on behalf of a given entity and can be addressed to another, acting as if it were an actual network. This will be leveraged to run software pre-packaged in the emulator, such as routing protocols that are essential for the correct expected behavior of a simulated network, but also to schedule software execution that was developed for this work, which includes the ALTO server, network state providers - e.g., probing daemons and application feedback collectors - and system clients for the P2P and HTTP mock applications that will be devised to play out a particular experiment scenario, and which will have embedded into it an ALTO client to interface with the server for council. As the simulation tool runs on Linux and builds a simulated network that behaves very much like a real one, well known real-application tools can be used on top of it in other needed areas, including the deployment and measuring phases, which gives plenty of flexibility on tool selection.

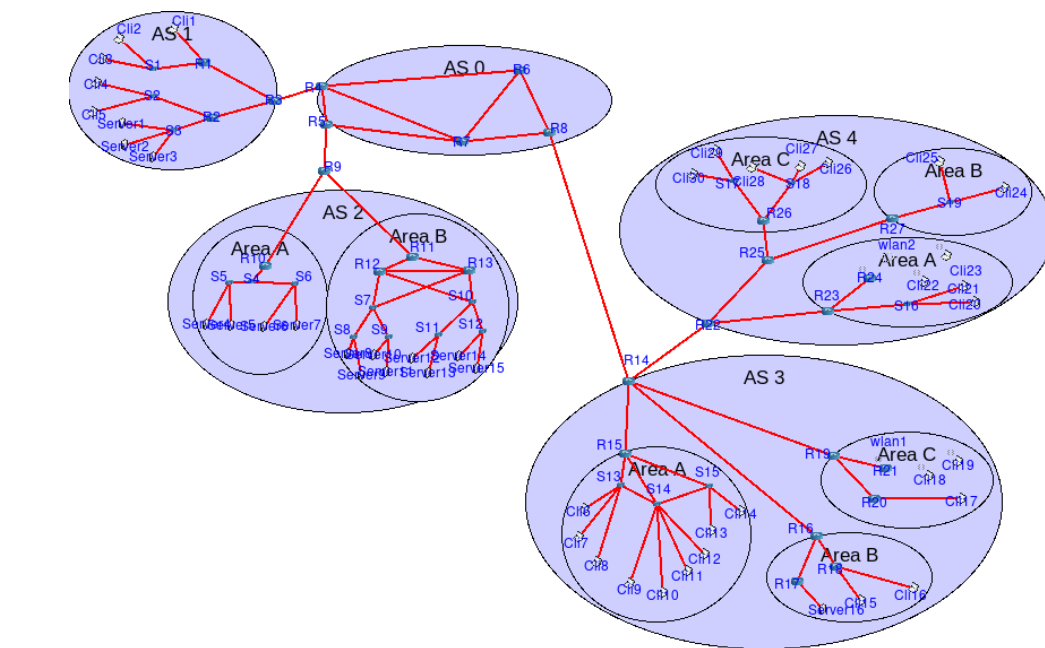
Python [?] will be utilized to implement all simple software prototypes whose purpose is uniquely to test the application in a real scenario. [?] This includes the P2P file-transfer applications, the HTTP servers and clients, and the throughput-intensive activities done by the data servers. Appended to this programming language will also be the task of application monitoring, which includes the retrieval of performance statistics - doing so in the application's code itself, instead of using external tools, because more fine grained access exists and individual tasks can be monitored for how long and how well they perform. The choice of this language over others is simply that these software prototypes are not intended to be highly optimized, nor are they to be complex. Instead, their mode of operation is supposed to be simple in nature, to remove complex variables that might make the experiment results harder to infer upon, and to make reasoning and replication of experiment results easier. Python seems then like a good fit due to its easy syntax, its interpreted nature that skips work that would otherwise be needed for compilation that might increase performance - but is not required - and, finally, its massive collection of helpful libraries. For these reasons will too Python be chosen to generate visual graphics reflective of the raw network and application statistics to be collected by other tools.

Finally, a tool was selected for the task of network monitoring, to collect network data representative of the impact that a given application strategy had towards the infrastructure. To this goal, vnStat [?] was chosen, a command line utility to measure network traffic on a per-interface basis, over given periods of time. This application retrieves network interface statistics provided directly by the kernel, and thus performs no traffic sniffing. This is not problematic as this level of detail is not needed for the designed experiments, and instead interface statistics that inform on data flow influx and outflux are sufficient. Finally, with vnStat being a command line utility, there is the additional bonus that deployment and orchestration are facilitated with scripting.

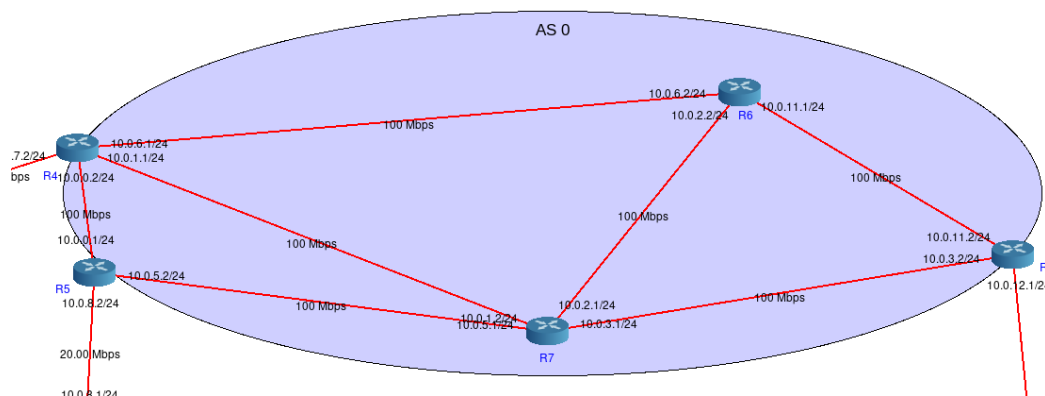
5.2 SETUP

Figure 5.1 displays the topology that will act as the main environment for all the devised experiments. It was designed with the intent of reflecting, at a smaller scale, the structure of the Internet, in particular with it being an aggregation of multiple, heterogeneous, domains, each with their own topological properties and internal policies, with them being administrated by different organizations. As can there be seen, a single backbone network - AS 0 - provides connectivity between many ASs and, to do

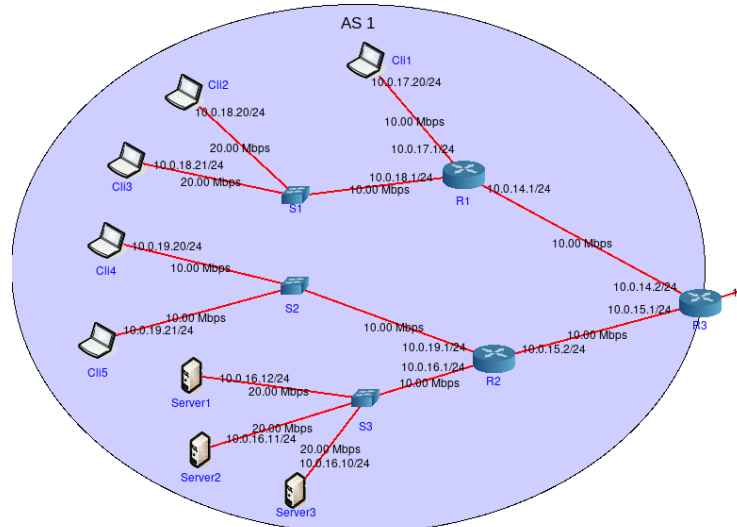
its job correctly, a high degree of path redundancy exists between its routers, and the links have better capabilities than those associated with stub networks. AS 1 is a simple topological structure consisting of five *Personal Computers (PCs)* and three dedicated servers, connected with the help of switches and routers, that eventually connect to a single edge router that connects to the backbone. AS 2 is representative of a data center with two OSPF areas, both constructed with a hierarchical organization common for data center networks. Links in these regions are also highly capable and high traffic peak times are expected to occur. AS 3 is a slightly more complex stub network compared to AS 1, but has the same structure, with the addition of having three OSPF areas instead of one, and a variety of nodes and links with different properties - for example, the links in area A are generally better, whilst area C has wireless connections in it that are expected to have worse performance and be less reliable. Finally, AS 4 connects directly with AS 3, meaning that the latter acts also as a transit AS, being the only access point towards the rest of the network. Similarly to AS 3, it consists of a stub network accessed by many end users and some servers, and both node and link properties vary accordingly.



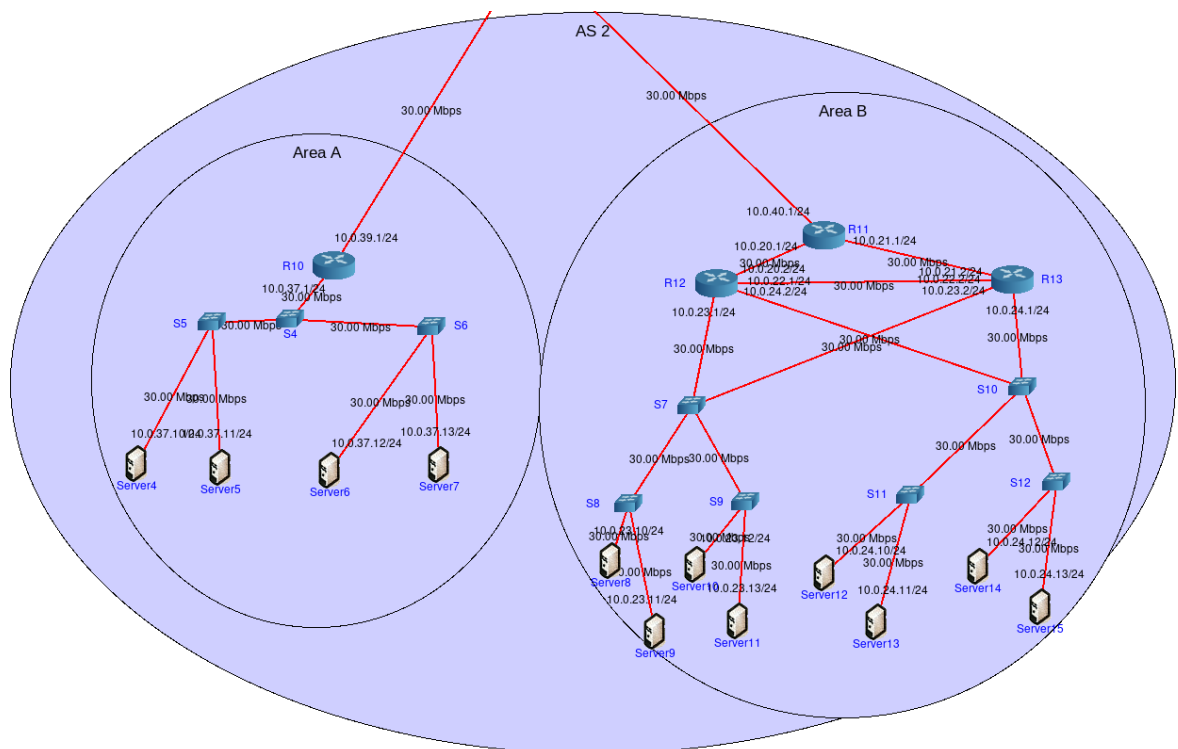
(a) Global topology view



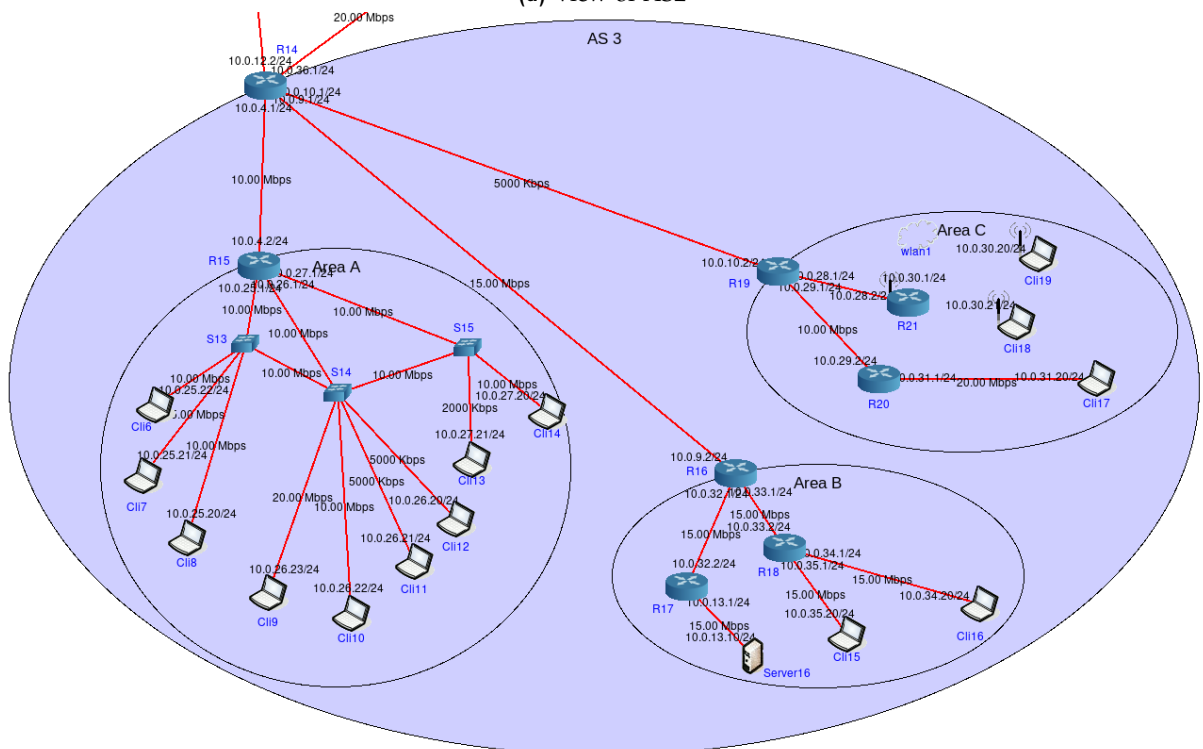
(b) View of AS0



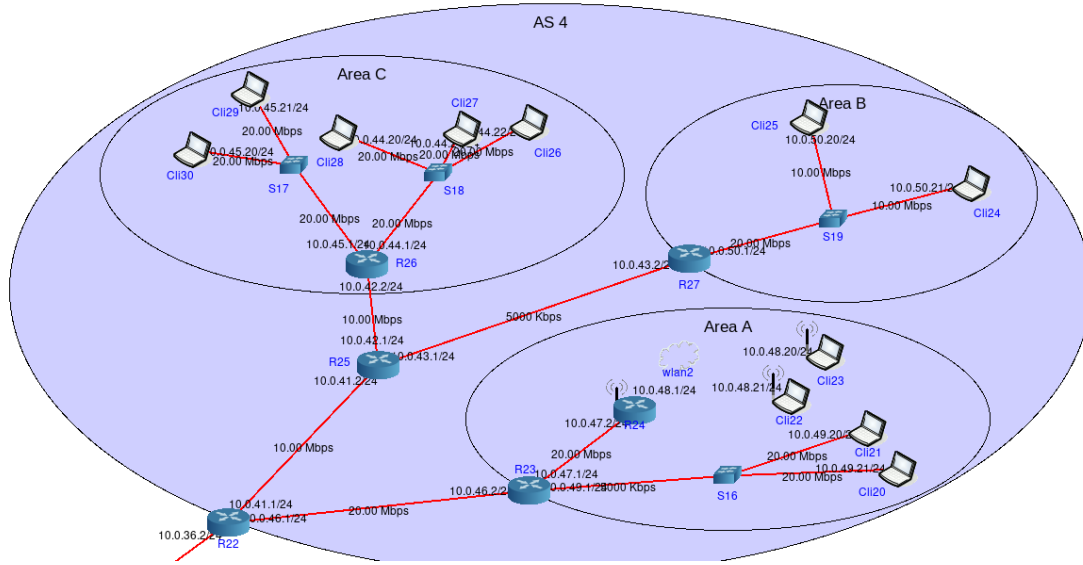
(c) View of AS1



(d) View of AS₂



(e) View of AS₃



(f) View of AS₄

Figure 5.1: values for different arms (cont.)

Each node on the network has a given purpose that is represented as a node label, and link labels are used to specify connection properties. Unless stated otherwise with these labels, all other properties are equal throughout the network. Some pre-packaged CORE services need to be enabled to assure network connectivity - mainly *Open Shortest Path First Version 3 (OSPFv3)* and BGP - and scripting is used to, at the beginning of the simulation, bootstrap programs in specific nodes. This is done by utilizing the command line utility `vcmd` that runs specified commands in control channels that are created at runtime by CORE - for example, the following command executes a ping command to address "10.0.0.1" with origins on node "P2P-Cli-1" and on simulation session 12345:

Listing 5.1: Execution of an example command through the control channel of a given node

```
$ vcmd -c /tmp/pycore.12345/P2P-Client-1 -- ping 10.0.0.1
```

5.3 SCENARIOS

5.3.1 Scenario 1 - P2P file sharing

Description: All "CliN" nodes in the network, with N from 1 to N, actively serve all ten equally sized fragments of a 1GB file to other peers, and the bootstrapping method includes informing the tracker about what file fragments they serve. 5.1 shows what endpoints possess what file fragment IDs.

Fragment ID	Available endpoints
x00	["10.0.18.21", "10.0.24.20", "10.0.35.20"],
x01	["10.0.18.20", "10.0.26.21", "10.0.48.21"],
x02	["10.0.19.21", "10.0.24.21", "10.0.35.20"],
x03	["10.0.24.21", "10.0.27.21", "10.0.31.20"],
x04	["10.0.18.21", "10.0.26.23", "10.0.24.20"],
x05	["10.0.19.20", "10.0.24.20", "10.0.24.21"],
x06	["10.0.27.21", "10.0.48.20", "10.0.49.21"],
x07	["10.0.48.21", "10.0.49.21", "10.0.50.21"],
x08	["10.0.50.20", "10.0.50.21", "10.0.35.20"],
x09	["10.0.25.22", "10.0.26.21", "10.0.31.20"]

Table 5.1: Tracker mappings of file fragments to available endpoints

"Cli2" wishes to retrieve that file, and to do so he firstly contacts "Tracker" for tracking information in the form of a file that contains a mapping between a fragment *Identifier (ID)* and the suggested peer, alongside with the file's checksum. After retrieving the mapping, the client will sequentially request each fragment from its appointed peer, finalizing with the merge all the fragments into a single file and calculating the checksum that will be compared with the one provided by the tracker. An ALTO server resides in the same AS and maintains resources - specifically, a local network map that groups peers within locality as within "Area A" of "AS₁", "Area B" of "AS₂", or externally to "AS₁". It also maintains global resources - specifically, an endpoint cost map indicating expected bandwidth and delay between all the peers participating in the overlay P2P network.

Variables: The variable actions to be tested are how the tracker selects what candidate peer, from the pool of the available ones, will be selected to serve a given file fragment to the requesting peer. Table 5.2 displays the different tracker algorithms that will be tested for the task of peer matching.

Tracker Algorithm	Description
Random	Randomly elect among all available peers
RTT	Select the peer with the smallest probe packet RTT measurement average in 10 pings
ALTO - Local	Retrieve the local network map from the requesting peer's ALTO server and discover the peers whose PID matches the requesting peer, choosing randomly if multiple options exist
ALTO - Global	Retrieve the global multcost endpoint cost map from the requesting peer's ALTO server by selecting the TCP throughput and one way delay metrics, and selecting the peer that maximizes throughput with a delay no bigger than 5 milliseconds.

Table 5.2: Tracker algorithms to be tested in scenario 1

Measurements: The measurements will target network resource usage and application performance. Table 5.3 presents the measurements that will be collected during the experiment runs.

Measurement	Units	Description
Transfer Time	Seconds	Total amount of time required to transfer all file fragments
Network Traffic	Megabytes	Total amount of traffic that entered each network area and AS

Table 5.3: Measurements to be taken in scenario 1

5.3.1.1 Scenario 2 - HTTP transfer scheduling

Description: "Server₁" in "AS₁" acts as a server of HTTP content. "Cli₁" wishes to retrieve a 500MB file from that server in one single session. The server is subjected to variable, random client loads that affect processing and storage power, and subsequently its ability to serve clients, as well as periodic traffic loads within the AS whenever the server clusters in that system exchange data between themselves for server redundancy and general synchronization. This will be translated in practice as dynamically variable round-trip delay from "Cli₁" to "Server₁", which is specifically shown in 5.4

Simulation time	Client-Server path delay
0 seconds	100 ms
60 seconds	50 ms
120 seconds	20 ms
180 seconds - ∞	10 ms

Table 5.4: Dynamically applied client-server path delays

The data center administrator maintains a cost map calendar where he registers all reserved backup transfers with two purposes: to keep track of all scheduled transfers so further ones can be scheduled as not to clash with pre-reserved ones, and to signal to ALTO clients when QoS levels to those servers are expected to degrade.

Variables: The variable action to be tested is how the HTTP client selects the most appropriate time to retrieve the content from the server. Table 5.5 displays the different client algorithms that will be tested for the task of server request scheduling.

Client Algorithm	Description
Immediate	Immediately request the resource from the server
RTT	Ping the server every 5 seconds and request the resource when the delay is below 20ms
ALTO	Retrieve a calendar cost map from the ALTO server in AS ₂ of expected path delay to the server, and initiate immediately when the delay is expected to drop below 20ms

Table 5.5: Client algorithms to be tested in scenario 2

Measurements: Table 5.6 presents the measurements that will be collected during the experiment runs.

Measurement	Units	Description
Transfer Time	Seconds	Total amount of time required to transfer the file
Network Traffic	Megabytes	Total amount of traffic that passed through each area network and AS

Table 5.6: Measurements to be taken in scenario 2

5.3.2 Scenario 3 - HTTP mirror selection

Description: An HTTP client wishes to retrieve a 500MB file from a server. After querying the public proxy, an index is provided which contains a address lists of the four mirror servers that provide that content. The listing is show in table 5.7

Server address	AS	Area
10.0.16.12	1	B
10.0.37.10	2	A
10.0.23.11	2	B
10.0.13.10	3	B

Table 5.7: Listed server mirrors

The mirror servers will be subjected to constant loads at the start of the scenario, that are translated as throughput throttling applied from the client to the servers, specifically the ones shown in 5.8.

Server address	Throughput
10.0.16.12	3.0 ms
10.0.37.10	5.0 ms
10.0.23.11	10.0 ms
10.0.13.10	5.0 ms

Table 5.8: Available path throughput from client to mirror servers

The ALTO server local to that client provides a global ALTO endpoint property map that contains CPU and RAM load information about all the mirrors, as well as an endpoint cost map containing information about the expected bandwidth and delay from the client to the mirror.

Variables: The variable action to be tested is how the HTTP client selects which mirror server to retrieve the file from. Table 5.9 displays the different client algorithms that will be tested for the task of mirror selection.

Client Algorithm	Description
Random	Randomly select a server
Bandwidth	Probe for available TCP throughput to the mirrors, and select the one with most bandwidth
ALTO	Retrieve an endpoint property map and endpoint cost map from the local ALTO server, and choose the with most available bandwidth and delay below 2ms, whose memory and processing power is below 30%

Table 5.9: Client algorithms to be tested in scenario 3

Measurements: Table 5.10 presents the measurements that will be collected during the experiment runs.

Measurement	Units	Description
Transfer Time	Seconds	Total amount of time required to transfer the file
Network Traffic	Megabytes	Total amount of traffic that passed through each area network and AS

Table 5.10: Measurements to be taken in scenario 3

5.4 RESULTS

This section will show the observed results for each of the simulated scenarios, highlighting how the different variable methods compared to each other.

5.4.1 Scenario 1

Figure 5.2 shows the obtained execution times for each of the variable methods in scenario 1. It appears that the method of probing for path delay was the worst, at 1096 seconds of runtime, closely followed by the method of randomly selecting between the available peers. Considerably faster are the ALTO-related options. The one considering a local server with its own information aided the application in achieving a 913.19 seconds execution time, whereas a global approach was able to considerably shave more time, at around 821.92 total seconds.

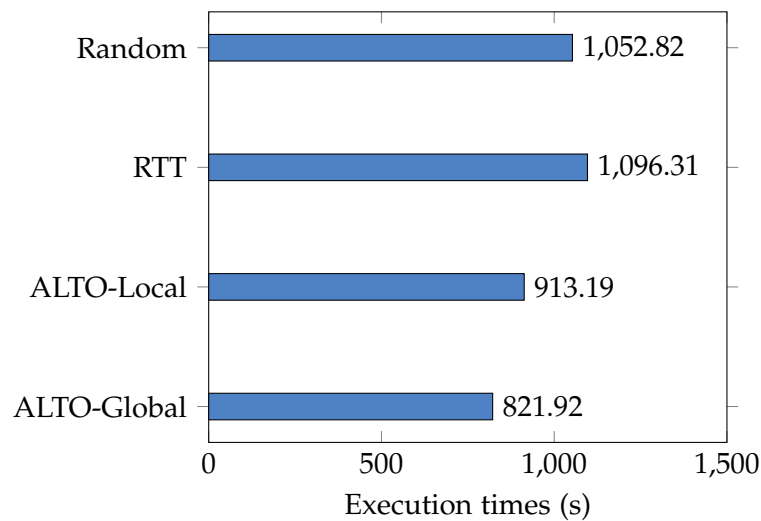


Figure 5.2: Execution times measured Scenario 1

Figure 5.3 show the measured traffic influx into the network's ASs and their areas. It can be immediately identified that the delay probing method incurred in considerable amount of traffic entering the local AS from the exterior. A random approach reduced that value almost in half, but the ALTO approaches were both equally proficient and had the lowest amount of incoming exterior traffic. The rest of the measurements was negligible in how tiny they were, with small bumps in "AS3" indicating choice preference for the delay-measuring method.

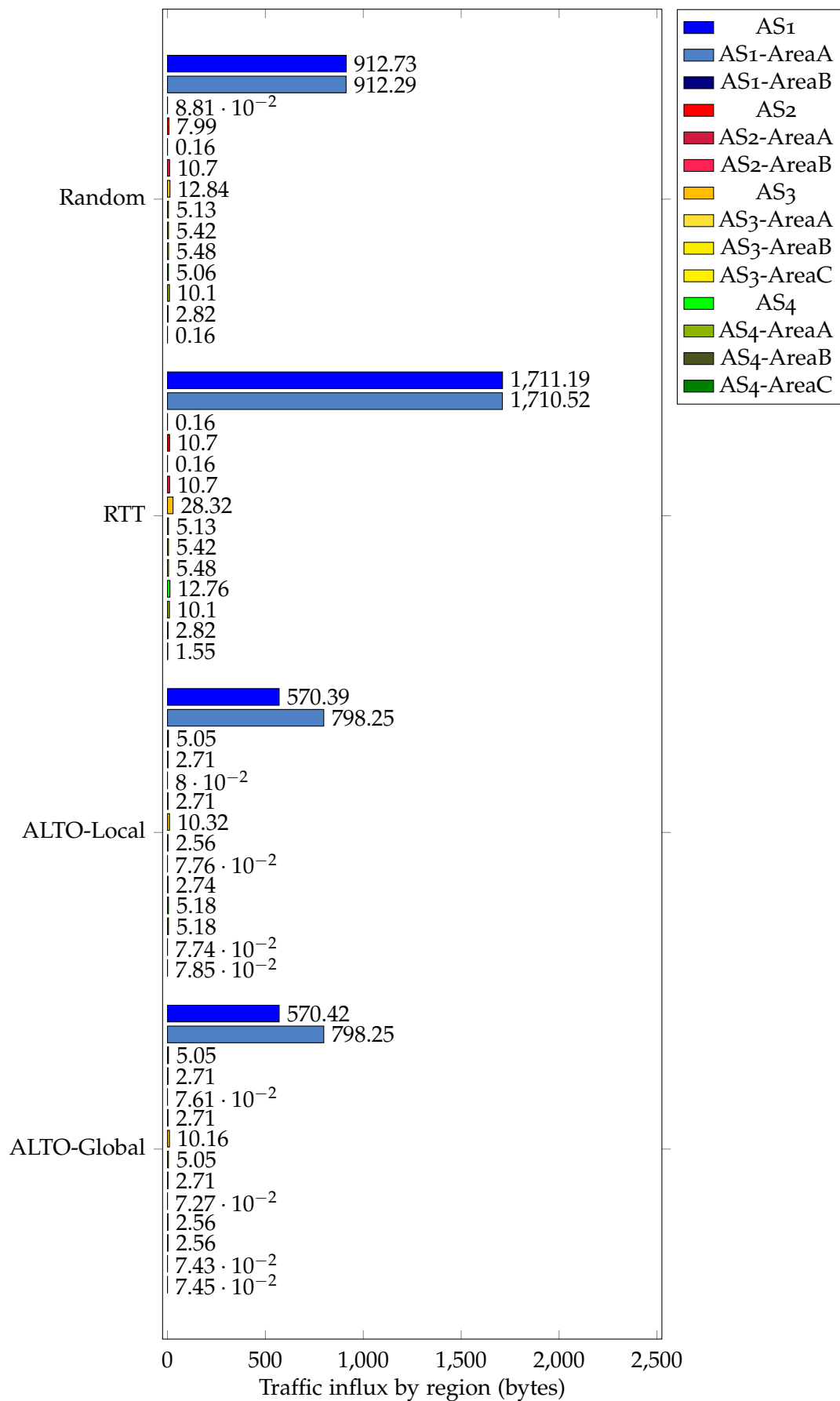


Figure 5.3: Inbound traffic flux by network areas measured in Scenario 1

5.4.2 Scenario 2

Figure 5.4 shows the obtained execution times for each of the variable methods in scenario 2. It appears that the ALTO-aided and delay-probing methods were equally circling a 640 second runtime, and an approach of immediately querying the server was faster, at 457.19 seconds.

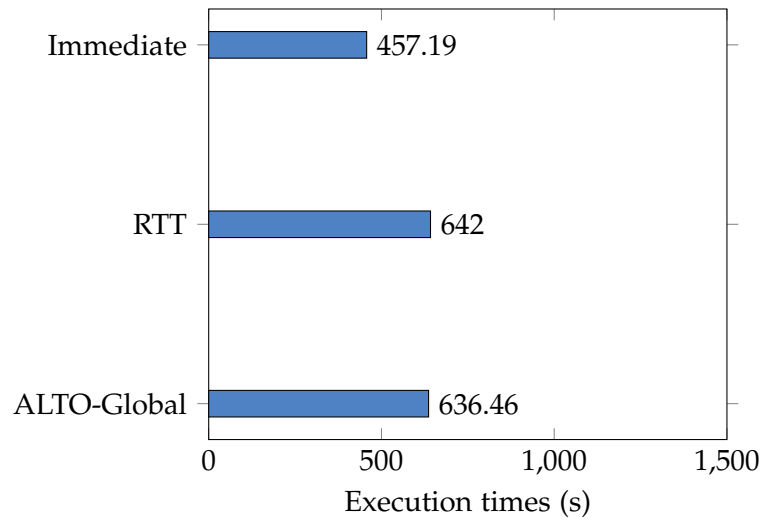


Figure 5.4: Execution times measured Scenario 2

Figure 5.5 shows the measured traffic influx into the network's ASs and their areas. Disregarding negligible measurements from given ASs and areas, it appears that the method utilizing ping measurements had slight, but existing, increased traffic that had its influx on AS1's areas.

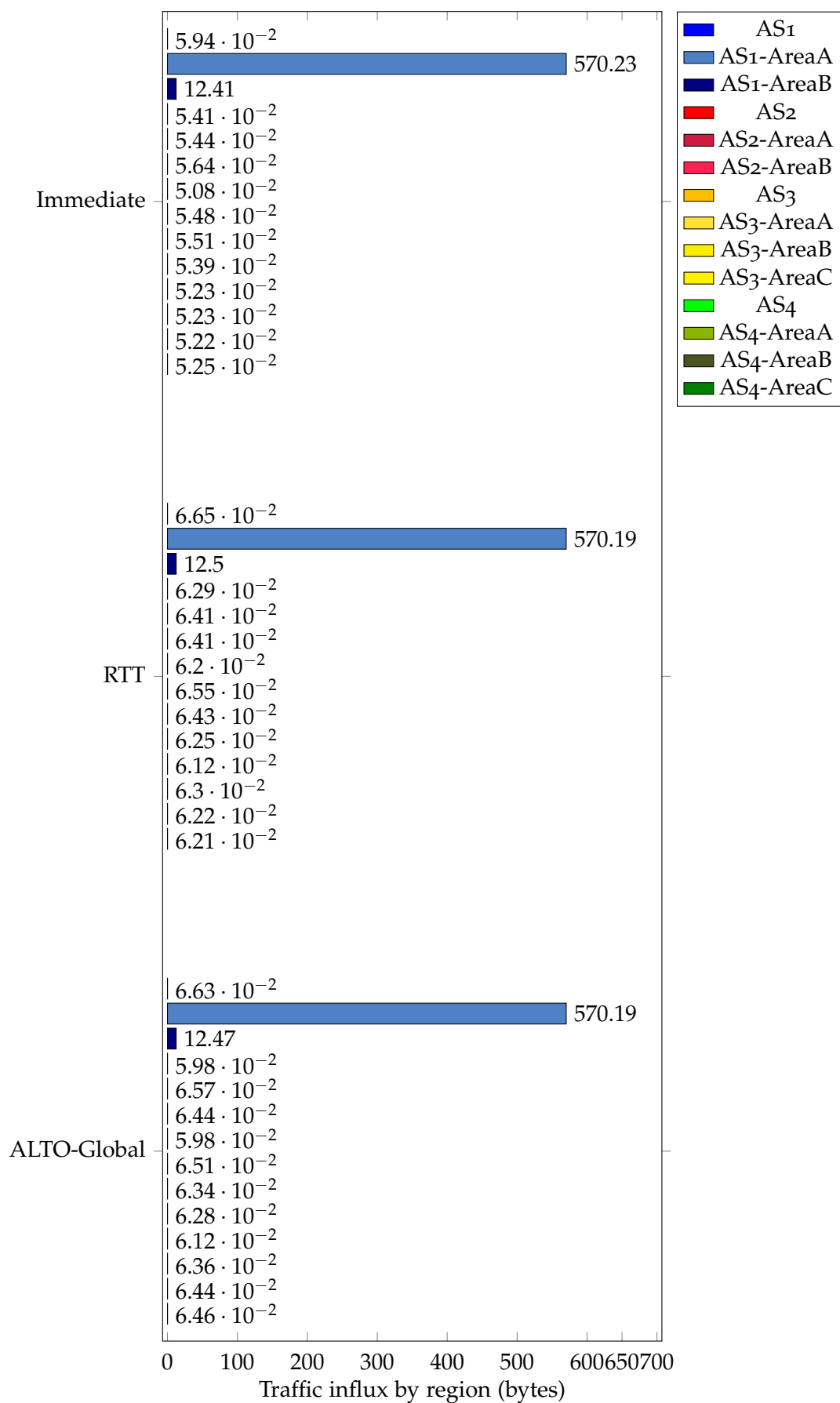


Figure 5.5: Inbound traffic flux by network areas measured in Scenario 2

5.4.3 Scenario 3

Figure 5.6 shows the obtained execution times for each of the variable methods in scenario 3. An approach of randomly choosing mirror servers had the worst performance time wise, with 912.39 seconds, and the TCP throughput measuring method following with 496.57 seconds. Finally, the ALTO-aided approach was the quickest with 456.21 seconds of runtime.

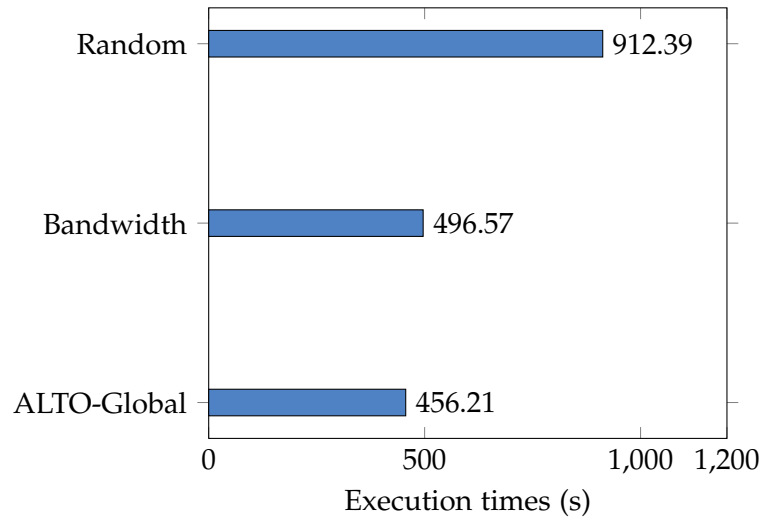


Figure 5.6: Execution times measured Scenario 3

Figure 5.7 shows the measured traffic influx into the network's ASs and their areas. It appears that the approach that probed for path bandwidth had a considerable impacts on the generated traffic influx, and the spikes of traffic in the random and ALTO approach give clues into the selected mirrors - those residing in AS₃-B and AS₂-B, most specifically.

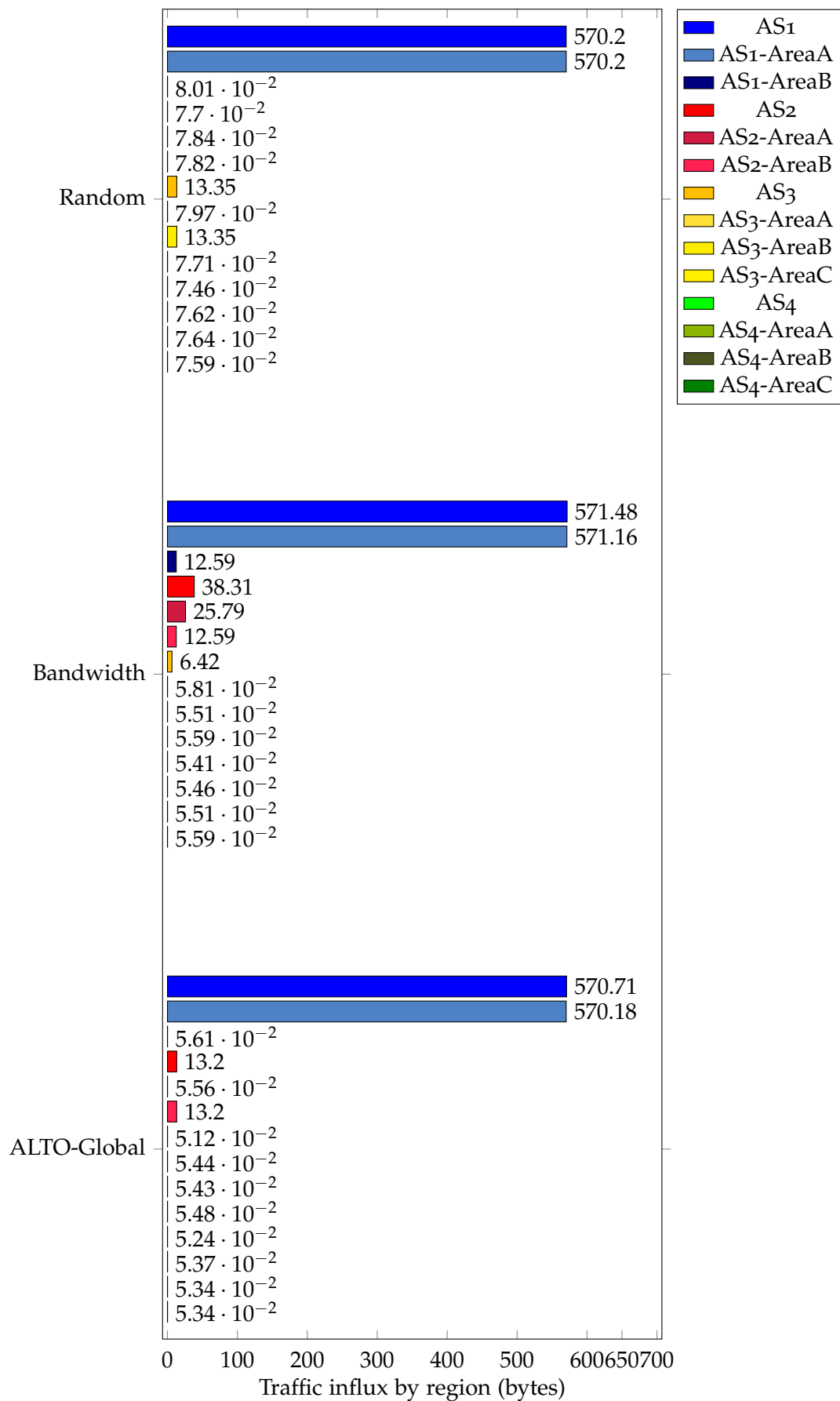


Figure 5.7: Inbound traffic flux by network areas measured in Scenario 3

5.5 DISCUSSION

[Notes: The P2P client has no reasonable way to, without ISP cooperation, know that peers reside within network locality. A cited study used RTT heuristics that assumed that high RTT values exist in inter-ISP links, which may not always be the case. Randomly selecting is poor at optimal choice, but at the very least has potential for load balancing, something that the tracker itself can attempt to do on a per-locality basis, joining a best of both worlds. Leveraging only a network map to communicate peer locality is quick and takes little memory, and further experiments will add cost maps on top of it to mix both locality standards and QoS needs.]

Finish discuss

6 | CONCLUSION

finish conclus

6.1 CONCLUSIONS

6.2 PROSPECT FOR FUTURE WORK

BIBLIOGRAPHY

- [1] Cisco visual networking index: Forecast and trends. Technical report, 2 2019.
- [2] The global internet phenomena report. Technical report, 9 2019.
- [3] Jan Seedorf, Sebastian Kiesel, and Martin Stiernerling. Traffic localization for p2p-applications: The alto approach. 10 2009.
- [4] Active network intelligence solutions | sandvine. <https://www.sandvine.com/>. Accessed: 2020-05-20.
- [5] Bittorrent.org. http://bittorrent.org/beps/bep_0003.html. Accessed: 2020-05-20.
- [6] Ppstream. <http://pps.tv/>. Accessed: 2020-05-20.
- [7] Akamai. <https://www.akamai.com/>. Accessed: 2020-09-20.
- [8] Erik Nygren, Ramesh Sitaraman, and Jennifer Sun. The akamai network: a platform for high-performance internet applications., 01 2010.
- [9] J. Liu, S. G. Rao, B. Li, and H. Zhang. Opportunities and challenges of peer-to-peer internet video broadcast. *Proceedings of the IEEE*, 96(1), 2008.
- [10] Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys (CSUR)*, 36, 12 2004.
- [11] Eng Lua, Jon Crowcroft, Marcelo Pias, Ravi Sharma, and Steven Lim. A survey and comparison of peer-to-peer overlay network schemes. *Communications Surveys Tutorials, IEEE*, 7:72– 93, 04 2006.
- [12] Regarding gnutella - gnu project - free software foundation. <https://www.gnu.org/philosophy/gnutella.en.html>. Accessed: 2020-05-20.
- [13] Wikipedia Commons. The gnutella search and retrieval protocol. <https://en.wikipedia.org/wiki/Gnutella#/media/File:GnutellaQuery.JPG>. Accessed: 2020-05-20.

- [14] Napster. <https://www.napster.com/>. Accessed: 2020-05-20.
- [15] Freenet. <https://freenetproject.org/>. Accessed: 2020-05-20.
- [16] I. Stoica, R. Morris, D. Liben-Nowell, D. R. Karger, M. F. Kaashoek, F. Dabek, and H. Balakrishnan. Chord: a scalable peer-to-peer lookup protocol for internet applications. *IEEE/ACM Transactions on Networking*, 11(1), 2003.
- [17] The free haven project. <https://www.freehaven.net/overview.html>. Accessed: 2020-05-20.
- [18] Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. *First Monday*, 5, 2000.
- [19] Claudio Fiandrino. P2p system topology. <https://texample.net/tikz/examples/p2p-topology/>. Accessed: 2020-06-04.
- [20] Q. Liao, Z. Li, and A. Striegel. Is more p2p always bad for isps? an analysis of p2p and isp business models. In *2014 23rd International Conference on Computer Communication and Networks (ICCCN)*, Aug 2014.
- [21] Aditya Akella, Srinivasan Seshan, and Anees Shaikh. An empirical evaluation of wide-area internet bottlenecks. *ACM SIGMETRICS Performance Evaluation Review*, 31, 05 2003.
- [22] Bram Cohen. Incentives build robustness in bittorrent. *Workshop on Economics of PeertoPeer systems*, 6, 06 2003.
- [23] F. Qin, J. Liu, L. Zheng, and L. Ge. An effective network-aware peer selection algorithm in bittorrent. In *2009 Fifth International Conference on Intelligent Information Hiding and Multimedia Signal Processing*, Sep. 2009.
- [24] J. H. Wang, D. M. Chiu, and J. C. s. Lui. Modeling the peering and routing tussle between isps and p2p applications. In *2006 14th IEEE International Workshop on Quality of Service*, 2006.
- [25] Thomas Karagiannis, Pablo Rodriguez, and Konstantina Papagiannaki. Should internet service providers fear peer-assisted content distribution? 01 2005.
- [26] Vinay Aggarwal, Stefan Bender, Anja Feldmann, and Arne Wichmann. Methodology for estimating network distances of gnutella neighbors. 01 2004.

- [27] György Dán, Tobias Hossfeld, Simon Oechsner, Piotr Cholda, Rafal Stankiewicz, Ioanna Papafili, and George Stamoulis. Interaction patterns between p2p content distribution systems and isps. *IEEE Communications Magazine*, 49, 05 2011.
- [28] Al-Mukaddim Khan Pathan, Rajkumar Buyya, and Design Computing. A taxonomy and survey of content delivery networks. 2006.
- [29] Evi Nemeth, Garth Snyder, Trent R. Hein, Ben Whaley, and Dan Mackin. *UNIX and Linux System Administration Handbook (5th Edition)*. Addison-Wesley Professional, 5th edition, 2017.
- [30] Netflix. <https://www.netflix.com>. Accessed: 2020-09-20.
- [31] Youtube. <https://www.youtube.com/>. Accessed: 2020-09-20.
- [32] Cloudflare. <https://www.cloudflare.com/>. Accessed: 2020-09-20.
- [33] Cloudfront. <https://aws.amazon.com/cloudfront/>. Accessed: 2020-09-20.
- [34] M. Wichtlhuber, J. Kessler, S. Bucker, I. Poesse, J. Blendin, C. Koch, and D. Hausheer. Soda: Enabling cdn-isp collaboration with software defined anycast. In *2017 IFIP Networking Conference (IFIP Networking) and Workshops*, June 2017.
- [35] Benjamin Frank, Ingmar Poesse, Yin Lin, Georgios Smaragdakis, Anja Feldmann, Bruce Maggs, Jannis Rake, Steve Uhlig, and Rick Weber. Pushing cdn-isp collaboration to the limit. *SIGCOMM Comput. Commun. Rev.*, 43(3):34–44, July 2013.
- [36] Raghunath Deshpande. Overview of cdn-isp collaboration strategies. 07 2014.
- [37] At&t. <https://www.att.com/>. Accessed: 2020-09-20.
- [38] Orange. <https://www.orange.com/>. Accessed: 2020-09-20.
- [39] Swisscom. <https://www.swisscom.ch/>. Accessed: 2020-09-20.
- [40] Kt. <https://corp.kt.com/>. Accessed: 2020-09-20.
- [41] Lu Liu and Nick Antonopoulos. *From Client-Server to P2P Networking*, pages 71–89. Springer US, 2010.
- [42] Linux mint. <https://linuxmint.com/>. Accessed: 2020-09-20.

- [43] Zahra Elngomi and Khalid Khanfar. A comparative study of load balancing algorithms: A review paper. In *International Journal of Computer Science and Mobile Computing*, pages 448–458, 06 2016.
- [44] Mei Chin, Chong Eng Tan, and Mohamad Bandan. Efficient load balancing for bursty demand in web based application services via domain name services. 01 2010.
- [45] Xiaohui Yang. Sdn load balancing method based on k-dijkstra. *International Journal of Performability Engineering*, 14, 04 2018.
- [46] Why you should switch to a different linux mint mirror today! <https://unlockforus.com/why-you-should-switch-to-a-different-linux-mint-mirror-today/>. Accessed: 2020-01-03.
- [47] Pedro Sousa. *Context Aware Programmable Trackers for the Next Generation Internet*, volume 5733, page 78. 2009.
- [48] Pedro Sousa. A framework for highly reconfigurable p2p trackers. *Journal of Communications Software and Systems*, 9(4):236, dec 2013.
- [49] D. Hughes, I. Warren, and G. Coulson. Agnus: the altruistic gnutella server. In *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003)*, 2003.
- [50] T. N. Kim, S. Jeon, and Y. Kim. A cdn-p2p hybrid architecture with content/location awareness for live streaming service networks. In *2011 IEEE 15th International Symposium on Consumer Electronics (ISCE)*, June 2011.
- [51] S. Ratnasamy, M. Handley, R. Karp, and S. Shenker. Topologically-aware overlay construction and server selection. In *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3, pages 1190–1199 vol.3, 2002.
- [52] Vinay Aggarwal and Anja Feldmann. Locality-aware p2p query search with isp collaboration. *NHM*, 3, 06 2008.
- [53] K. Han, Q. Guo, and J. Luo. Optimal peer selection, task assignment and rate allocation for p2p downloading. In *2009 First International Workshop on Education Technology and Computer Science*, volume 1, March 2009.

- [54] M. L. Gromov and Y. P. Chebotareva. On optimal cdn node selection. In *2014 15th International Conference of Young Specialists on Micro/Nanotechnologies and Electron Devices (EDM)*, June 2014.
- [55] Ben Niven-Jenkins, François Le Faucheur, and Dr. Nabil N. Bitar. Content Distribution Network Interconnection (CDNI) Problem Statement. RFC 6707, September 2012.
- [56] P. Francis, S. Jamin, Cheng Jin, Yixin Jin, D. Raz, Y. Shavitt, and L. Zhang. Idmaps: a global internet host distance estimation service. *IEEE/ACM Transactions on Networking*, 9(5):525–540, 2001.
- [57] T. S. E. Ng and Hui Zhang. Predicting internet network distance with coordinates-based approaches. In *Proceedings.Twenty-First Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1, pages 170–179 vol.1, 2002.
- [58] I. Poesse, B. Frank, B. Ager, G. Smaragdakis, S. Uhlig, and A. Feldmann. Improving content delivery with padis. *IEEE Internet Computing*, 16(3):46–52, 2012.
- [59] Benjamin Frank, Ingmar Poesse, Georgios Smaragdakis, Steve Uhlig, and Anja Feldmann. Content-aware traffic engineering. *CoRR*, abs/1202.1464, 2012.
- [60] K. Mase, A. Tsuno, Y. Toyama, and N. Karasawa. A web server selection algorithm using qos measurement. In *ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No.01CH37240)*, volume 8, June 2001.
- [61] M. Swain and Young-Gyun Kim. Finding an optimal mirror site. In *Proceedings. IEEE SoutheastCon, 2005.*, April 2005.
- [62] André Sampaio and Pedro Sousa. An adaptable and ISP-friendly multicast overlay network. *Peer-to-Peer Networking and Applications*, 12(4):809–829, September 2018.
- [63] Application-layer traffic optimization (alto). Technical report, 11 2019.
- [64] Jan Seedorf, Y. Richard Yang, Kevin J. Ma, Jon Peterson, Xiao Shawn Lin, and Jingxuan Jensen Zhang. Content Delivery Network Interconnection (CDNI) Request Routing: CDNI Footprint and Capabilities Advertisement using ALTO. Internet-Draft draft-ietf-alto-cdni-request-routing-alto-08, Internet Engineering Task Force, November 2019. Work in Progress.

- [65] Luis M. Contreras, Danny Alex Lachos Perez, and Christian Esteve Rothenberg. Use of ALTO for Determining Service Edge. Internet-Draft draft-contreras-alto-service-edge-01, Internet Engineering Task Force, July 2020. Work in Progress.
- [66] Kai Gao, Young Lee, Sabine Randriamasy, Y. Richard Yang, and J. (Jensen) Zhang. ALTO Extension: Path Vector. Internet-Draft draft-ietf-alto-path-vector-11, Internet Engineering Task Force, July 2020. Work in Progress.
- [67] Sabine Randriamasy, Y. Richard Yang, Qin Wu, Deng Lingli, and Nico Schwan. Application-Layer Traffic Optimization (ALTO) Cost Calendar. Internet-Draft draft-ietf-alto-cost-calendar-21, Internet Engineering Task Force, March 2020. Work in Progress.
- [68] Sebastian Kiesel, Wendy Roome, Richard Woundy, Stefano Previdi, Stanislav Shalunov, Richard Alimi, Reinaldo Penno, and Y. Richard Yang. Application-Layer Traffic Optimization (ALTO) Protocol. RFC 7285, September 2014.
- [69] Jan Seedorf and Eric Burger. Application-Layer Traffic Optimization (ALTO) Problem Statement. RFC 5693, October 2009.
- [70] Martin Stiernerling, Sebastian Kiesel, Michael Scharf, Hans Seidel, and Stefano Previdi. Application-Layer Traffic Optimization (ALTO) Deployment Considerations. RFC 7971, October 2016.
- [71] The stride threat model. [https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878\(v=cs.20\)?redirectedfrom=MSDN](https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)?redirectedfrom=MSDN). Accessed: 2020-09-20.
- [72] Barbara Van Schewick. Network neutrality and quality of service: What a non-discrimination rule should look like. Technical report, 6 2012.
- [73] Federal communications commission. <https://www.fcc.gov/>. Accessed: 2020-09-20.
- [74] Regulation (eu) 2015/2120 of the european parliament and of the council. <https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32015R2120&rid=2#d1e445-1-1>. Accessed: 2020-09-20.
- [75] Plusnet. <https://www.plus.net/>. Accessed: 2020-09-20.

- [76] Nate Anderson. Deep packet inspection meets ‘net neutrality’. <https://arstechnica.com/gadgets/2007/07/deep-packet-inspection-meets-net-neutrality/2/>. Accessed: 2020-09-20.
- [77] Meo. <https://www.meo.pt/>. Accessed: 2020-09-20.
- [78] Tarifários móveis pós-pagos unlimited. <https://www.meo.pt/telemovel/tarifarios/unlimited>. Accessed: 2017-12-14.
- [79] Facebook. <https://www.facebook.com/>. Accessed: 2020-09-20.
- [80] Spotify. <https://www.spotify.com/>. Accessed: 2020-09-20.
- [81] Wikipedia zero. https://en.wikipedia.org/wiki/Wikipedia_Zero. Accessed: 2020-09-20.
- [82] Wikipedia. <https://en.wikipedia.org>. Accessed: 2020-09-20.
- [83] Lily Hay Newman. Net neutrality is already in trouble in the developing world. <https://slate.com/technology/2014/01/net-neutrality-internet-access-is-already-in-trouble-in-the-developing-world.html>, 1 2014. Accessed: 2020-25-10.
- [84] J. Domzal, R. Wójcik, and A. Jajszczyk. Qos-aware net neutrality. In *2009 First International Conference on Evolving Internet*, pages 147–152, 2009.
- [85] Danny Alex Lachos Perez, Christian Esteve Rothenberg, Qiao Xiang, Y. Richard Yang, Börje Ohlman, Sabine Randriamasy, Farni Boten, Luis M. Contreras, J. (Jensen) Zhang, and Kai Gao. Supporting Multi-domain Use Cases with ALTO. Internet-Draft draft-lachos-alto-multi-domain-use-cases-01, Internet Engineering Task Force, July 2020. Work in Progress.
- [86] Qin Wu, Y. Richard Yang, Young Lee, Dhruv Dhody, and Sabine Randriamasy. ALTO Performance Cost Metrics. Internet-Draft draft-ietf-alto-performance-metrics-08, Internet Engineering Task Force, November 2019. Work in Progress.
- [87] Sabine Randriamasy, Wendy Roome, and Nico Schwan. Multi-Cost Application-Layer Traffic Optimization (ALTO). RFC 8189, October 2017.
- [88] Kimo Bumanglag and Houssain Kettani. On the impact of dns over https paradigm on cyber systems. pages 494–499, 03 2020.

- [89] Java. <https://www.java.com/en/>. Accessed: 2020-09-20.
- [90] Java spring. <https://spring.io/>. Accessed: 2020-09-20.
- [91] Mongodb. <https://www.mongodb.com/>. Accessed: 2020-09-20.
- [92] Julian Reschke. The 'Basic' HTTP Authentication Scheme. RFC 7617, September 2015.
- [93] Rifaat Shekh-Yusef, David Ahrens, and Sophie Bremer. HTTP Digest Access Authentication. RFC 7616, September 2015.

