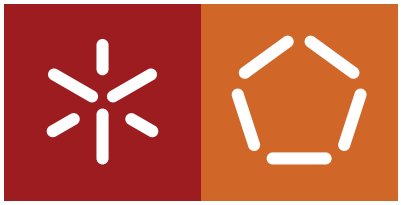
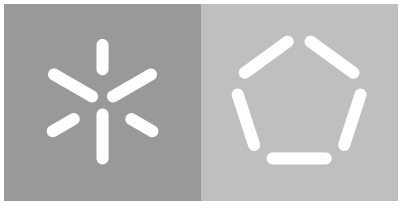


Universidade do Minho
Escola de Engenharia
Departamento de Informática

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

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 estandardizações 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 xiii
List of Tables xiv
List of Acronyms xv
1 EXPERIMENTS 1
1.1 Technologies Used . . . 2
1.2 Setup . . . 3
1.3 Scenarios . . . 7
1.3.1 Scenario 1 - P2P file sharing . . . 7
1.3.1.1 Overview . . . 7
1.3.1.2 Analysis of Results . . . 9
1.3.2 Scenario 2 - HTTP transfer scheduling . . . 12
1.3.2.1 Overview . . . 12
1.3.2.2 Analysis of Results . . . 13
1.3.3 Scenario 3 - HTTP mirror selection . . . 15
1.3.3.1 Overview . . . 15
1.3.3.2 Analysis of Results . . . 16
1.4 Discussion . . . 19

Bibliography 21

LIST OF FIGURES

Figure 1.1	Network topology and integrated *Autonomous Systems (ASs)* . .	7
Figure 1.2	Execution times measured Scenario 1	10
Figure 1.3	Inbound traffic flux by network areas measured in Scenario 1 .	11
Figure 1.4	Execution times measured Scenario 2	13
Figure 1.5	Inbound traffic flux by network areas measured in Scenario 2 .	14
Figure 1.6	Execution times measured Scenario 3	17
Figure 1.7	Inbound traffic flux by network areas measured in Scenario 3 .	18

LIST OF TABLES

Table 1.1	Tracker mappings of file fragments to available endpoints	8
Table 1.2	Tracker algorithms to be tested in scenario 1	9
Table 1.3	Measurements to be taken in scenario 1	9
Table 1.4	Dynamically applied client-server path delays	12
Table 1.5	Client algorithms to be tested in scenario 2	12
Table 1.6	Measurements to be taken in scenario 2	13
Table 1.7	Listed server mirrors	15
Table 1.8	Available path throughput from client to mirror servers	15
Table 1.9	Client algorithms to be tested in scenario 3	16
Table 1.10	Measurements to be taken in scenario 3	16

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.
FTP File Transfer Protocol.

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.
() .

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 EXPERIMENTS

The purpose of this chapter is to overview the experimentation phase of the project, which contains the work done to deploy and measure how applications behave when using either classic or *Application-Layer Traffic Optimization (ALTO)*-guided solutions when faced with some of the constructed scenarios, that force some decision which impacts the underlying network. 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 correctly the system behaves .

Adjacent to the goal of testing the system's behavior in an emulated environment, the experimentation phase also aims to embed in such environment a list of case study scenarios where applications could leverage the ALTO system to their advantage, and subsequently observe and measure if and how the ALTO server can help the client with its network insight 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 Chapter **??**, together with the specified system extensions on Chapter **??** leads one to believe that a theoretical mutually beneficial scenario exists in an ALTO approach that could not exist in one where only one of the layers gets all the input when applying traffic optimization decisions. 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. In general, this discussion benefits the ALTO project and can give more maturity to the system as it was put through multiple test scenarios against other common strategies.

Section 1.1 displays the chosen technologies for tasks pertaining to the experiments. Section 1.2 focuses on the required steps taken to setup the testing environment. This includes the design and deployment of a network topology to be emulated, the creation of mock applications to serve as clients for the system, and the design and deployment of application and network status measurement tools. Section 1.3 follows with the individual overview of the devised scenarios to be executed, 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, Section 1.3.3.2 will display the obtained results that were collected emulated environment, and Section 1.4 after that will discuss these results and how they fare with the theoretical findings.

1.1 TECHNOLOGIES USED

The *Common Open Research Emulator (CORE)* [98] was used as a network emulator 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 maximum 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 *Peer-to-Peer (P2P)* and *Hypertext Transfer Protocol (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.

vcmd

Python [99] 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 [100] 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.

1.2 SETUP

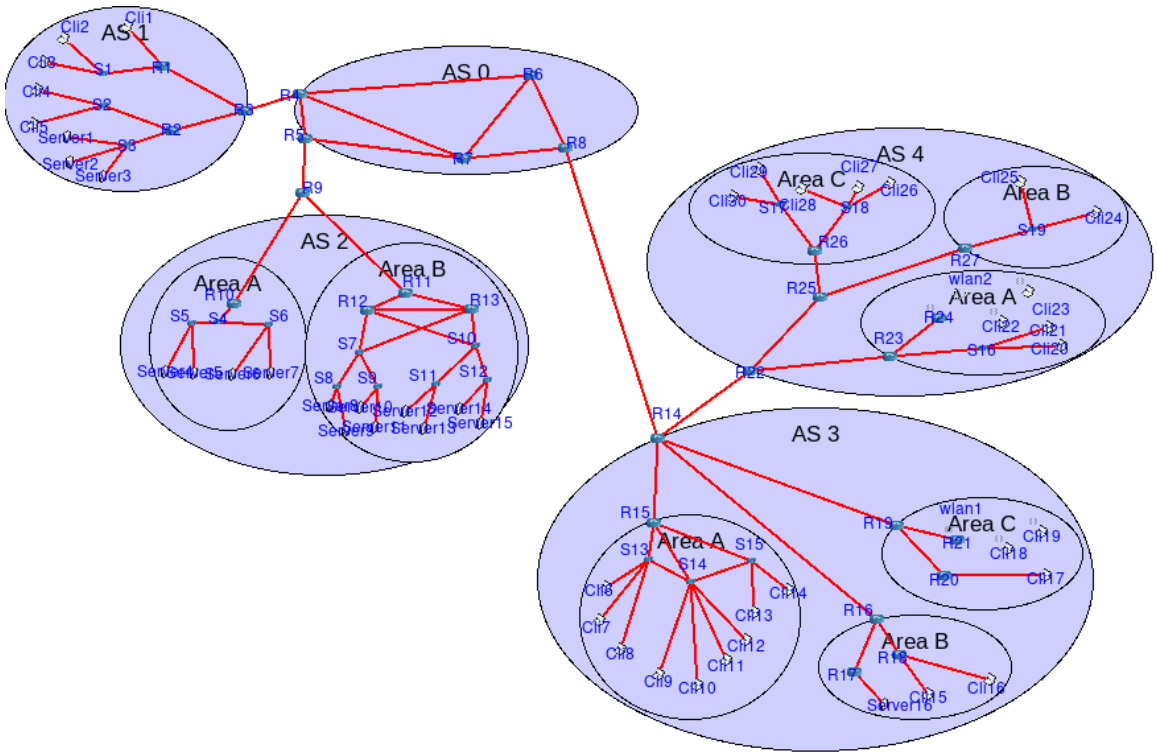
Figure 1.1 displays the topology that will act as the main environment for all the devised experiments, with partitioned views to be subsequently introduced. Figure 1.1a

presents a simplified global view the network as a collection of *Autonomous Systems (ASs)*. 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. A single backbone network, shown on Figure 1.1b, 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. Figure 1.1c shows AS 1, 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 links with the backbone. Figure 1.1d shows AS 2, which is representative of a data center with two *Open Shortest Path First (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, shown in Figure 1.1e, 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, depicted on Figure 1.1f, connects directly with AS 3, meaning that it also acts as a transit AS to 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.

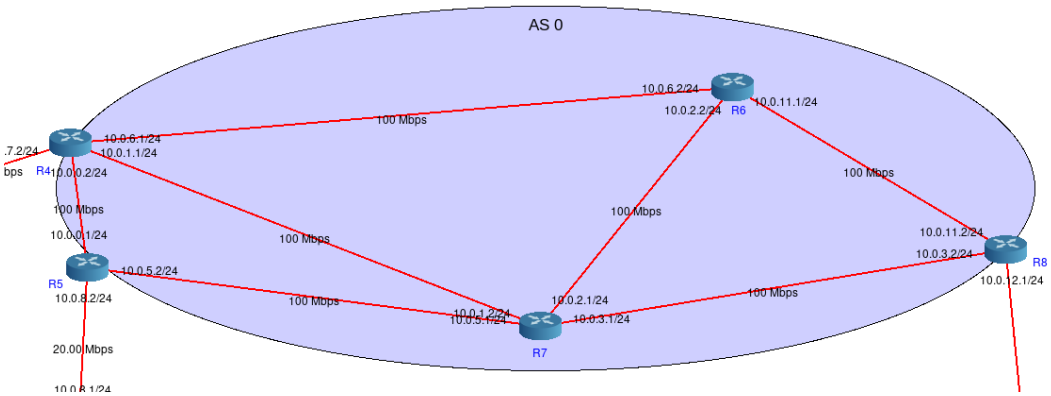
Each node on the network has a given purpose that is represented as a node label, and link labels are used to specify connection properties. Unless stated otherwise with these labels, all other properties are equal throughout the network. Some prepackaged CORE services need to be enabled to assure network connectivity - mainly *Open Shortest Path First Version 2 (OSPFv2)* and *Border Gateway Protocol (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 1.1: Execution of an example command through the control channel of a given node

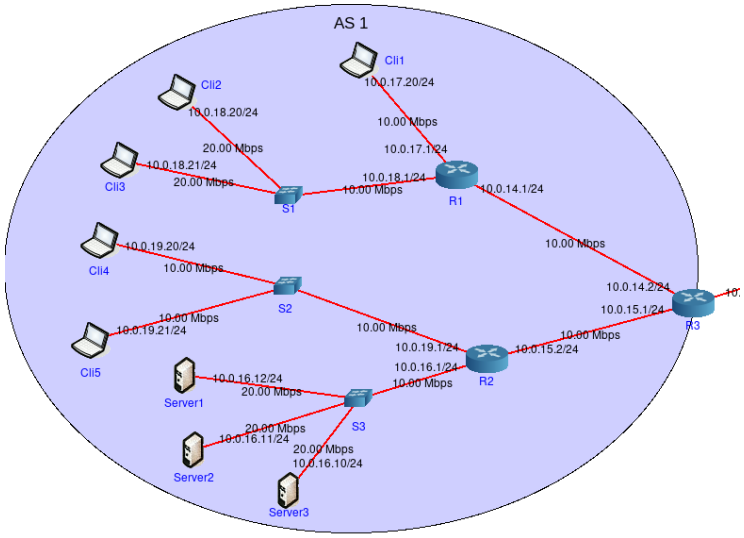
```
$ vcmd -c /tmp/pycore.12345/P2P-Client-1 -- ping 10.0.0.1
```



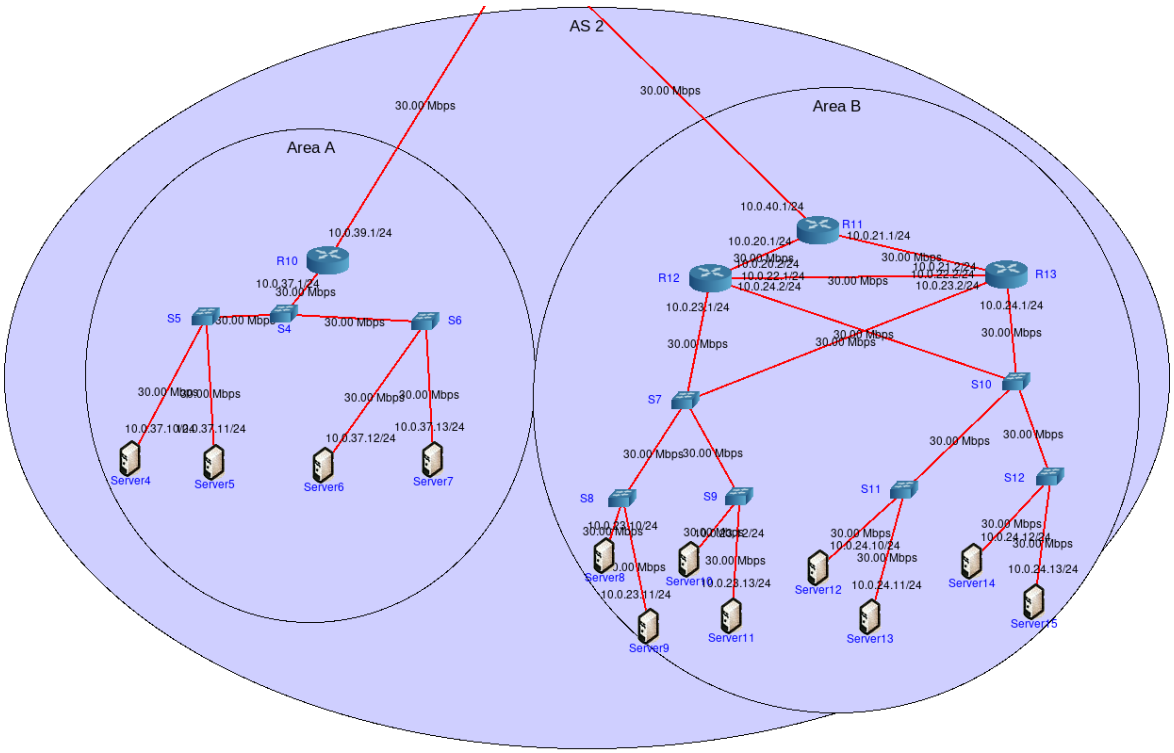
(a) Global view of the network



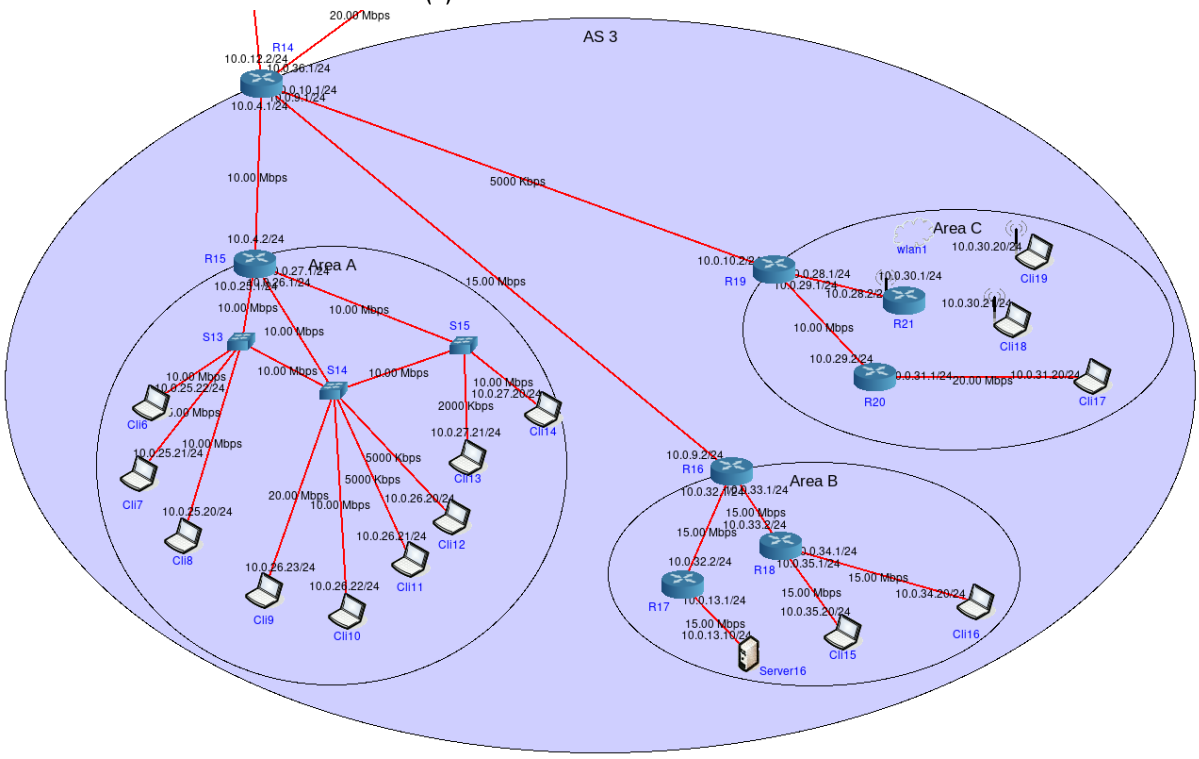
(b) Detailed view of AS0



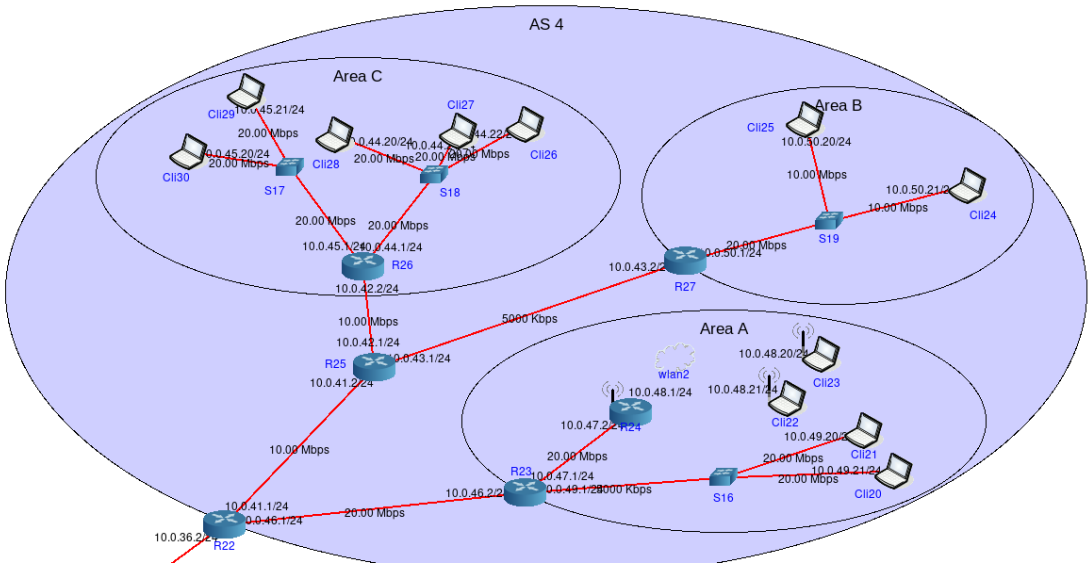
(c) Detailed view of AS1



(d) Detailed view of AS2



(e) Detailed view of AS3



(f) Detailed view of AS4

Figure 1.1: Network topology and integrated ASs

1.3 SCENARIOS

This Section describes the three devised scenarios to test the impact to the overlay and underlay that results of applications performing traffic optimization decisions with and without the implemented ALTO solution. Section 1.3.1 focuses on a P2P application who needs to decide, whenever a file fragment is served by multiple peers, with whom to establish connection. Section 1.3.2 tackles the issue of an HTTP server subjected to periodic load spikes, and how clients can selectively choose the timing to initialize the resource request. Section 1.3.3 deals with the challenge of selecting a given HTTP server to retrieve a content from whenever a mirror cluster is available.

1.3.1 Scenario 1 - P2P file sharing

1.3.1.1 Overview

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 pers, and the bootstrapping method includes informing the tracker about what file fragments they serve. 1.1 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 1.1: 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 retrieveing the mapping, the client will sequencially 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 1.2 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 *Round-Trip Time (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 *Provider-Defined Identifier (PID)* matches the requesting peer, choosing randomly if multiple options exist
ALTO - Global	Retrieve the global multicost endpoint cost map from the requesting peer's ALTO server by selecting the *Transmission Control Protocol (TCP)* throughput and one way delay metrics, and selecting the peer that maximizes throughput with a delay no bigger than 5 milliseconds.

Table 1.2: Tracker algorithms to be tested in scenario 1

Measurements: The measurements will target network resource usage and application performance. Table 1.3 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 1.3: Measurements to be taken in scenario 1

1.3.1.2 Analysis of Results

Figure 1.2 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.

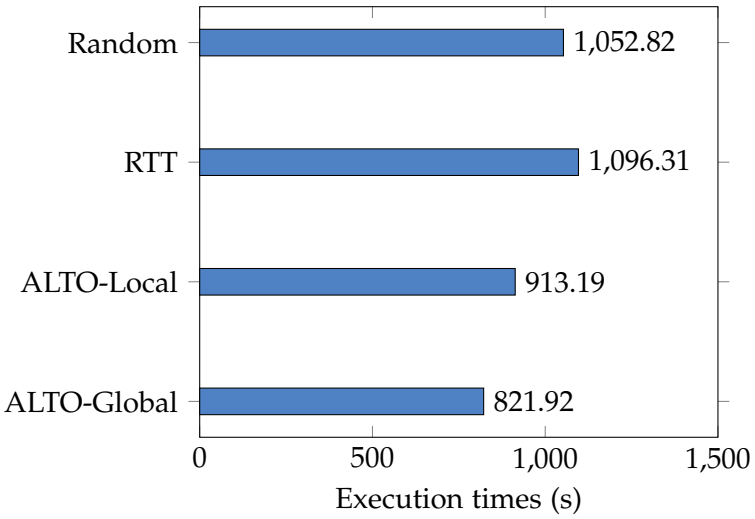


Figure 1.2: Execution times measured Scenario 1

Figure 1.3 show the measured traffic influx into the network's ASs and their areas. It can be immediatelly 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.

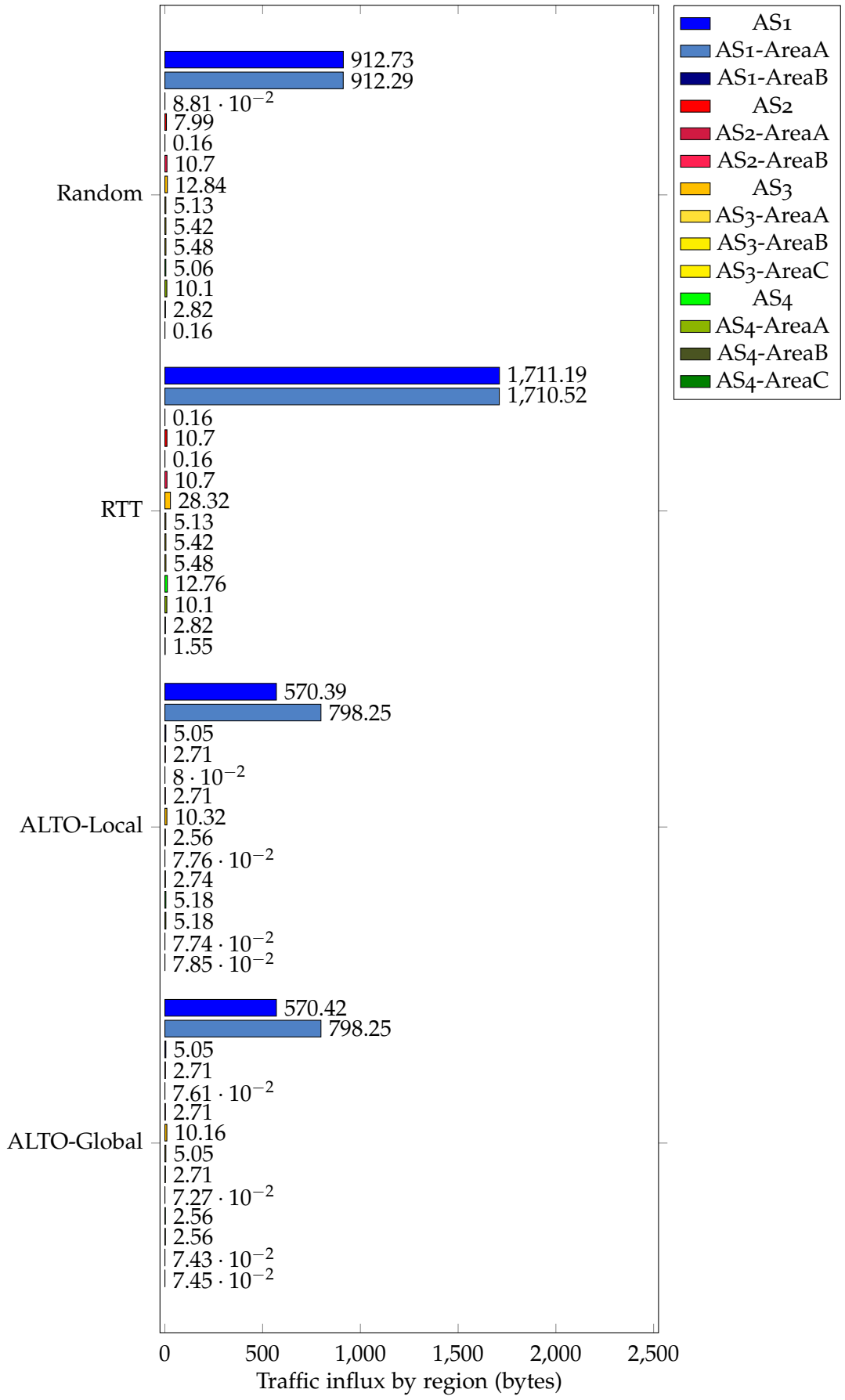


Figure 1.3: Inbound traffic flux by network areas measured in Scenario 1

1.3.2 Scenario 2 - HTTP transfer scheduling

1.3.2.1 Overview

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 enchange 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 1.4

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

Table 1.4: Dynamically applied client-server path delays

The data center administrator mantains 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 *Quality of Service (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 1.5 displays the different client algorithms that will be tested for the task of server request scheduling.

Client Algorithm	Description
Immediate	Immediatelly 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 AS2 of expected path delay to the server, and iniciate immediately when the delay is expected to drop below 20ms

Table 1.5: Client algorithms to be tested in scenario 2

Measurements: Table 1.6 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 1.6: Measurements to be taken in scenario 2

1.3.2.2 Analysis of Results

Figure 1.4 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.

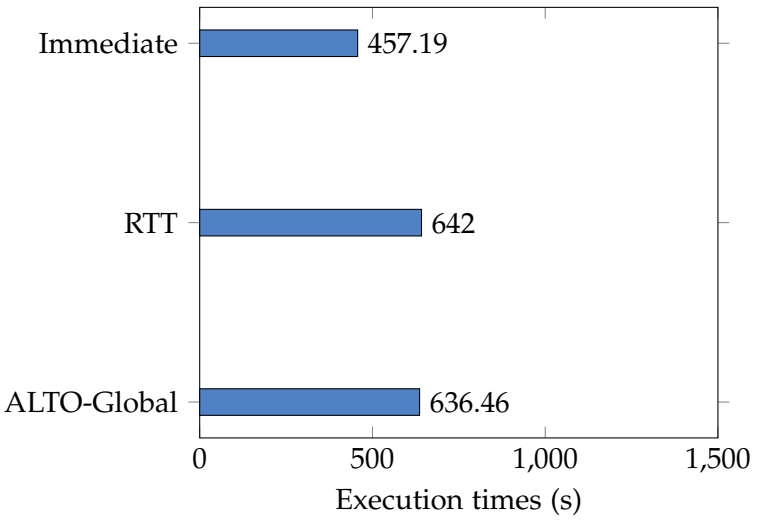


Figure 1.4: Execution times measured Scenario 2

Figure 1.5 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.

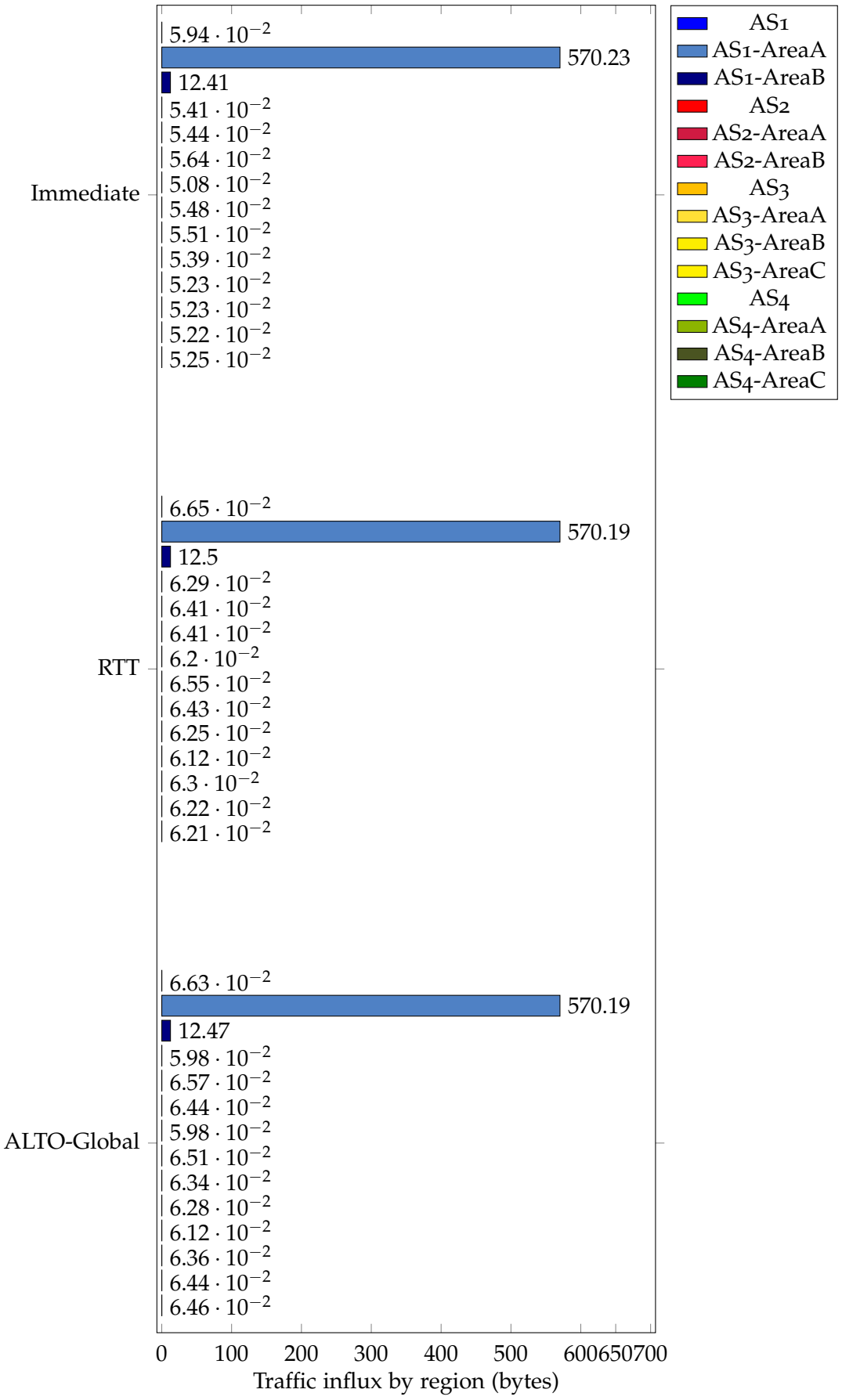


Figure 1.5: Inbound traffic flux by network areas measured in Scenario 2

1.3.3 Scenario 3 - HTTP mirror selection

1.3.3.1 Overview

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 1.7

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 1.7: 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 1.8.

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 1.8: Available path throughput from client to mirror servers

The ALTO server local to that client provides a global ALTO endpoint property map that contains *Central Processing Unit (CPU)* and *Random-Access Memory (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 1.9 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 1.9: Client algorithms to be tested in scenario 3

Measurements: Table 1.10 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 1.10: Measurements to be taken in scenario 3

1.3.3.2 Analysis of Results

Figure 1.6 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.

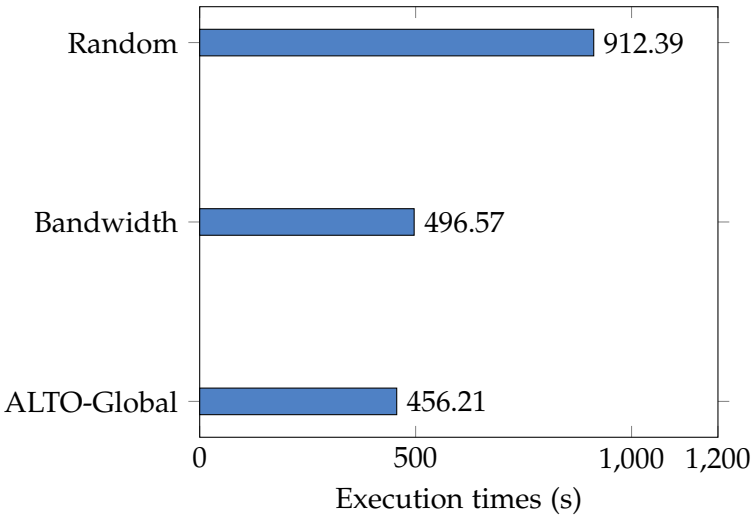


Figure 1.6: Execution times measured Scenario 3

Figure 1.7 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.

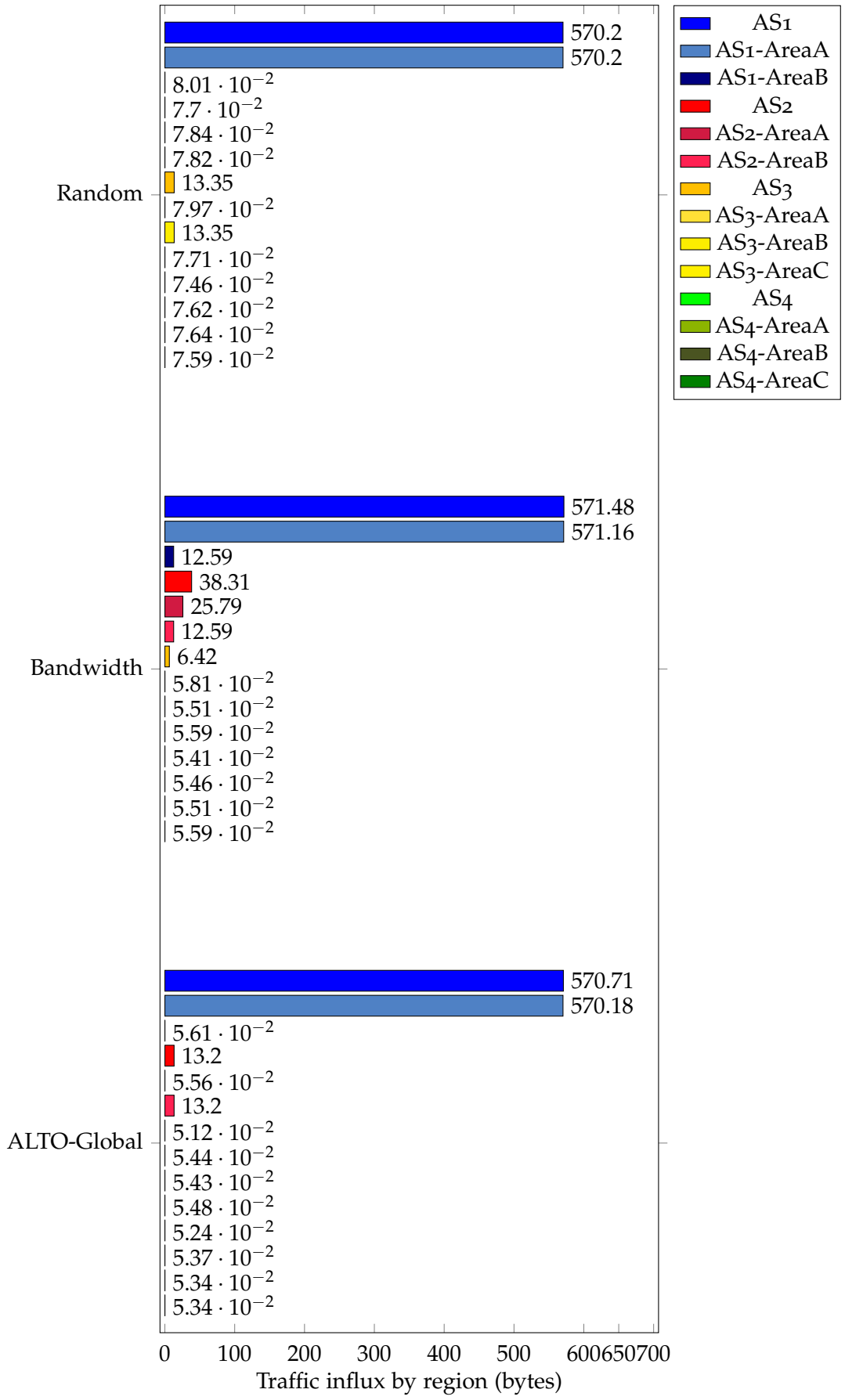


Figure 1.7: Inbound traffic flux by network areas measured in Scenario 3

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

1.4 DISCUSSION

The results seem to support the main advantages associated with the ALTO approach.

Regarding Scenario 1, it seems like a random selection of peers, as was expected, was far from optimal for execution time. Whilst random selection can be thought of as a good strategy to guarantee an even load distribution between peers, acting purely with this goal in mind is clearly not efficient in regards to network resources, as peers with worst network conditions can be chosen, and traffic can more often escape outside of network regions, as indeed was shown on the measurement of traffic that entered AS1, that meant a considerably amount of peers were selected outside of network regions. Perhaps surprisingly, a method of peer selection based on probing measurements targeting packet delay performed worse in regards to execution time and traffic locality. It appears that, for this topology and this overlay network, message delay was a bad indicator of application speed performance, and it favored a lot of peers that resided outside of local regions. The ALTO method revealed to be the most efficient at both runtime and traffic localization. A local approach was simple to implement, required input from only one *Internet Service Provider (ISP)* knowledge domain, and was able to get good time efficiency and very good traffic localization. Its knowledge gaps came from situations where all candidate peers for a given fragment resided externally, and the global approach that united knowledge from multiple ALTO domains helped pick those with a better throughput, additionally optimizing execution time.

Regarding scenario 2, immediately querying the server resulted in faster time, but this required that the server be queried during heavy load, possibly pushing the server over its capacity and reducing the ability to serve the current users whilst blocking potential candidates. A method of periodically querying the server with ping messages made it so the query initialized when the server load was greatly reduced, but this was done with a slight but noticeable amount of traffic generated. This was due to the fact that ping measurements, while simple in what information can be derived, incur in small traffic overhead, and the more information is attempted to be derived from the network the higher the overhead penalization is to be expected. With no surprise, an ALTO approach was able to, with a single query message to the ALTO

server, schedule when its file request should execute, obtaining a runtime similar to the probing approach but without probing traffic overhead.

Regarding scenario 3, it seems like the random mirror selection had the worst runtime, probably due to the fact that most of the servers were under load, and randomly selecting the best one, over many runs, was unlikely. A throughput probing approach was able to select the most apt mirror server to provide the server, but at the expense of much traffic overhead required to deduce available throughput. Additionally, this approach required from all mirror servers the hosting of a probing server that made these measurements possible, a luxury not possible in much real-case scenarios. The ALTO approach was able to get immediate ISP input in the form of a combined multi-ALTO domain effort, and quickly query the optimal mirror, resulting in a slightly better runtime than a probing approach.

BIBLIOGRAPHY

[1] Cisco visual networking index: Forecast and trends. Technical report, 2 2019.

[2] Vitor Pereira, Miguel Rocha, Paulo Cortez, Miguel Rio, and Pedro Sousa. A framework for robust traffic engineering using evolutionary computation. In *7th International Conference on Autonomous Infrastructure, Management and Security (AIMS 2013)*, LNCS 7943, pages 2–13, Barcelona, Spain, 2013. Springer.

[3] V. Pereira, M. Rocha, and P. Sousa. Traffic engineering with three-segments routing. *IEEE Transactions on Network and Service Management*, 17(3):1896–1909, 9 2020.

[4] The global internet phenomena report. Technical report, 9 2019.

[5] Jan Seedorf, Sebastian Kiesel, and Martin Stiemerling. Traffic localization for p2p-applications: The alto approach. 10 2009.

[6] Active network intelligence solutions | sandvine. `https://www.sandvine.com/`. Accessed in: 2020-05-20.

[7] Bittorrent.org. `http://bittirrent.org/beps/bep_0003.html`. Accessed in: 2020-05-20.

[8] Ppstream. `http://pps.tv/`. Accessed in: 2020-05-20.

[9] Akamai. `https://www.akamai.com/`. Accessed in: 2020-09-20.

[10] Erik Nygren, Ramesh Sitaraman, and Jennifer Sun. The akamai network: a platform for high-performance internet applications., 01 2010.

[11] 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.

[12] Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Computing Surveys (CSUR)*, 36, 12 2004.

[13] 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.

[14] Regarding gnutella - gnu project - free software foundation. `https://www.gnu.org/philosophy/gnutella.en.html`. Accessed in: 2020-05-20.

[15] Wikipedia Commons. The gnutella search and retrieval protocol. `https://en.wikipedia.org/wiki/Gnutella#/media/File:GnutellaQuery.JPG`. Accessed in: 2020-05-20.

[16] Napster. `https://www.napster.com/`. Accessed in: 2020-05-20.

[17] Freenet. `https://freenetproject.org/`. Accessed in: 2020-05-20.

[18] 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.

[19] The free haven project. `https://www.freehaven.net/overview.html`. Accessed in: 2020-05-20.

[20] Eytan Adar and Bernardo A. Huberman. Free riding on gnutella. *First Monday*, 5, 2000.

[21] Claudio Fiandrino. P2p system topology. `https://texample.net/tikz/examples/p2p-topology/`. Accessed in: 2020-06-04.

[22] 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.

[23] Aditya Akella, Srinivasan Seshan, and Anees Shaikh. An empirical evaluation of wide-area internet bottlenecks. *ACM SIGMETRICS Performance Evaluation Review*, 31, 05 2003.

[24] Bram Cohen. Incentives build robustness in bittorrent. *Workshop on Economics of PeertoPeer systems*, 6, 06 2003.

[25] 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.

[26] J. H. Wang, D. M. Chiu, and J. C. s. Lui. Modeling the peering and routing tussle between isps and p2p applications. In *200614th IEEE International Workshop on Quality of Service*, 2006.

[27] Thomas Karagiannis, Pablo Rodriguez, and Konstantina Papagiannaki. Should internet service providers fear peer-assisted content distribution? 01 2005.

[28] Vinay Aggarwal, Stefan Bender, Anja Feldmann, and Arne Wichmann. Methodology for estimating network distances of gnutella neighbors. 01 2004.

[29] 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.

[30] Al-Mukaddim Khan Pathan, Rajkumar Buyya, and Design Computing. A taxonomy and survey of content delivery networks. 2006.

[31] 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.

[32] Netflix. `https://www.netflix.com`. Accessed in: 2020-09-20.

[33] Youbtube. `https://www.youtube.com/`. Accessed in: 2020-09-20.

[34] Cloudflare. `https://www.cloudflare.com/`. Accessed in: 2020-09-20.

[35] Cloudfront. `https://aws.amazon.com/cloudfront/`. Accessed in: 2020-09-20.

[36] M. Wichtlhuber, J. Kessler, S. Bücker, I. Poese, 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.

[37] Benjamin Frank, Ingmar Poese, 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.

[38] Raghunath Deshpande. Overview of cdn-isp collaboration strategies. 07 2014.

[39] At&t. `https://www.att.com/`. Accessed in: 2020-09-20.

[40] Orange. `https://www.orange.com/`. Accessed in: 2020-09-20.

[41] Swisscom. https://www.swisscom.ch/. Accessed in: 2020-09-20.

[42] Kt. https://corp.kt.com/. Accessed in: 2020-09-20.

[43] Lu Liu and Nick Antonopoulos. *From Client-Server to P2P Networking*, pages 71–89. Springer US, 2010.

[44] Linux mint. https://linuxmint.com/. Accessed in: 2020-09-20.

[45] 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.

[46] 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.

[47] Xiaohui Yang. Sdn load balancing method based on k-dijkstra. *International Journal of Performability Engineering*, 14, 04 2018.

[48] 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 in: 2020-01-03.

[49] Pedro Sousa. *Context Aware Programmable Trackers for the Next Generation Internet*, volume 5733, page 78. 2009.

[50] Pedro Sousa. A framework for highly reconfigurable p2p trackers. *Journal of Communications Software and Systems*, 9(4):236, dec 2013.

[51] Pedro Sousa. Towards effective control of p2p traffic aggregates in network infrastructures. *Journal of Communications Software and Systems*, 11:37–47, 04 2015.

[52] D. Hughes, I. Warren, and G. Coulson. Agnus: the altruistic gnutella server. In *Proceedings Third International Conference on Peer-to-Peer Computing (P2P2003)*, 2003.

[53] 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.

[54] 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.

[55] Vinay Aggarwal and Anja Feldmann. Locality-aware p2p query search with isp collaboration. *NHM*, 3, 06 2008.

[56] 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.

[57] 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.

[58] Ben Niven-Jenkins, François Le Faucheur, and Dr. Nabil N. Bitar. Content Distribution Network Interconnection (CDNI) Problem Statement. RFC 6707, September 2012.

[59] 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.

[60] 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.

[61] I. Poese, B. Frank, B. Ager, G. Smaragdakis, S. Uhlig, and A. Feldmann. Improving content delivery with padis. *IEEE Internet Computing*, 16(3):46–52, 2012.

[62] Benjamin Frank, Ingmar Poese, Georgios Smaragdakis, Steve Uhlig, and Anja Feldmann. Content-aware traffic engineering. *CoRR*, abs/1202.1464, 2012.

[63] 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.

[64] M. Swain and Young-Gyun Kim. Finding an optimal mirror site. In *Proceedings. IEEE SoutheastCon, 2005.*, April 2005.

[65] 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.

[66] Application-layer traffic optimization (alto). Technical report, 11 2019.

[67] 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.

[68] 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.

[69] 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.

[70] 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.

[71] 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.

[72] 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.

[73] Jan Seedorf and Eric Burger. Application-Layer Traffic Optimization (ALTO) Problem Statement. RFC 5693, October 2009.

[74] Martin Stiemerling, Sebastian Kiesel, Michael Scharf, Hans Seidel, and Stefano Previdi. Application-Layer Traffic Optimization (ALTO) Deployment Considerations. RFC 7971, October 2016.

[75] The stride threat model. https://docs.microsoft.com/en-us/previous-versions/commerce-server/ee823878(v=cs.20)?redirectedfrom=MSDN. Accessed in: 2020-09-20.

[76] Barbara Van Schewick. Network neutrality and quality of service: What a non-discrimination rule should look like. Technical report, 6 2012.

[77] Federal communications commission. https://www.fcc.gov/. Accessed in: 2020-09-20.

[78] 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 in: 2020-09-20.

[79] Plusnet. https://www.plus.net/. Accessed in: 2020-09-20.

[80] Nate Anderson. Deep packet inspection meets 'net neutrality'. https://arstechnica.com/gadgets/2007/07/deep-packet-inspection-meets-net-neutrality/2/. Accessed in: 2020-09-20.

[81] Meo. https://www.meo.pt/. Accessed in: 2020-09-20.

[82] Tarifários móveis pós-pagos unlimited. https://www.meo.pt/telemovel/tarifarios/unlimited. Accessed in: 2017-12-14.

[83] Facebook. https://www.facebook.com/. Accessed in: 2020-09-20.

[84] Spotify. https://www.spotify.com/. Accessed in: 2020-09-20.

[85] Wikipedia zero. https://en.wikipedia.org/wiki/Wikipedia_Zero. Accessed in: 2020-09-20.

[86] Wikipedia. https://en.wikipedia.org. Accessed in: 2020-09-20.

[87] 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 in: 2020-25-10.

[88] J. Domzal, R. Wójcik, and A. Jajszczyk. Qos-aware net neutrality. In *2009 First International Conference on Evolving Internet*, pages 147–152, 2009.

[89] 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.

[90] Sabine Randriamasy, Wendy Roome, and Nico Schwan. Multi-Cost Application-Layer Traffic Optimization (ALTO). RFC 8189, October 2017.

[91] Kimo Bumanglag and Houssain Kettani. On the impact of dns over https paradigm on cyber systems. pages 494–499, 03 2020.

[92] Java. `https://www.java.com/en/`. Accessed in: 2020-09-20.

[93] Java spring. `https://spring.io/`. Accessed in: 2020-09-20.

[94] Mongodb. `https://www.mongodb.com/`. Accessed in: 2020-09-20.

[95] Julian Reschke. The 'Basic' HTTP Authentication Scheme. RFC 7617, September 2015.

[96] Rifaat Shekh-Yusef, David Ahrens, and Sophie Bremer. HTTP Digest Access Authentication. RFC 7616, September 2015.

[97] Prabath Siriwardena. Http basic/digest authentication. In *Advanced API Security*, pages 33–46.

[98] Common open research emulator. `https://www.nrl.navy.mil/itd/ncs/products/core`. Accessed in: 2020-09-20.

[99] Python. `https://www.python.org/`. Accessed in: 2020-09-20.

[100] vnstat. `https://humdi.net/vnstat/`. Accessed in: 2020-09-20.