Technische Universität München
Department of Informatics

Bachelor's Thesis in Informatics

Optimization of Quality Criteria Through Usage of Overlay Networks on the Internet

Stefan Fochler

Technische Universität München

Department of Informatics

Bachelor's Thesis in Informatics

Optimization of Quality Criteria Through Usage of Overlay Networks on the Internet

Optimierung von Qualitätskriterien durch den Einsatz von Overlay-Netzen im Internet

Author	Stefan Fochler
Supervisor	Prof. Dr.-Ing. Georg Carle
Advisor	Dipl.-Inf. Stephan-A. Posselt
	Dipl.-Inf. Johann Schlamp
Date	August 15th 2015

Informatik VIII
Chair for Network Architectures and Services

I confirm that this thesis is my own work and I have documented all sources and material used.

Garching b. München, August 15th 2015

Signature

Abstract

Overlay networks are used to deliver personalized contents over the Internet in situations where techniques such as caching are not possible due to the kind of data being transmitted. By measuring and comparing routes in the Internet, it is often times possible to find better routes featuring lower round trip times and therefore also enabling higher throughputs.

This work uses the PlanetLab project to instrument distributed measurements on up to 100 nodes doing round trip times measurements for regular Internet routes and routes through the Akamai overlay network. This network, even though only using links of the public Internet, promises improvements over the regular Internet route where possible by selecting the ideal route.

It is concluded that the overlay network manages to improve round trip times by up to 15 % (40 ms) for nodes with a very large distance to the web server. Measurement nodes that are connected using an already optimal route, on the other side, only experience an overhead of up to 5 ms of additional round trip time because of the processing time required by the Akamai Servers.

Finally, case studies examine both transport ways for three exemplary routes and analyze the different paths taken and the general differences for various cases of overlay routing acceleration. Interestingly, the case studies find an example where a route with 32 IP hops features lower round trip times than its plain Internet counterpart with only 15 hops.

Zusammenfassung

Overlay Netzwerke werden im kommerziellen Rahmen dazu eingesetzt, um personalisierte Inhalte schneller über das Internet auszuliefern, wenn auf Grund der Beschaffenheit der Daten kein caching durchführbar ist. Durch das Vermessen und Vergleichen von Routen im Internet kann in vielen Fällen eine bessere Route gefunden werden, die die Round-Trip Zeit reduziert und dadurch schnellere Übertragungen ermöglicht.

Diese Arbeit nutzt das PlanetLab Projekt, um von bis zu 100 verteilten Messknoten aus Zeitmessungen über reguläres Internet Routing und das Akamai Overlay Netzwerk durchzuführen. Letzteres nutzt zwar ausschließlich öffentliche Internet Verbindungen, verspricht aber durch die Wahl der optimalen Route eine Beschleunigung gegenüber der regulären Route.

Es wird festgestellt, dass das das Overlay Netzwerk bis zu 15 % (40 ms) kürzere Round-Trip Zeiten für Messknoten mit sehr großer Distanz zum Web Server ermöglicht. Messknoten mit bereits optimaler Anbindung über das normale Internet Routing hingegen benötigen bei Nutzung des Overlay Netzes bis zu 5 ms zusätzliche Zeit durch die Verarbeitung der Packete durch Akamai Server.

Abschließend werden für drei exemplarische Routen beide Transportwege in Einzelanylsen verglichen und verschiedene Fälle der Beschleunigung durch Overlay Netze sichtbar gemacht. Das deutlichste Beispiel ist dabei eine Overlay Netzwerk Route mit 32 IP Hops, die bessere Round-Trip Zeiten bietet als ihr Gegenstück über das normale Internet mit nur 15 Hops.

Acknowledgements

I would like to express my gratitude to my supervisors Dipl.-Inf. Johann Schlamp, Dipl.-Inf. Stephan A. Posselt and M.Sc. Nadine Herold who have spend many hours guiding me through the learning process of this bachelor thesis. They always kept their ears open and critically challenged my results in order to ensure their correctness.

I also would like to thank my father Manfred Fochler and his supervisor Ralf Gehrke at Akamai Technologies GmbH in Munich for making this thesis possible. By not only providing access to the services necessary, but also spending time for configuration and evaluation of the technical setup, they played an important role in enabling the measurements needed.

Furthermore, I would like to thank all the staff at chair 8 at the Technische Universität München with special focus on Prof. Dr.-Ing. Georg Carle for accepting this thesis in the first place, and the systems administrator Dipl.-Inform. Andreas Korsten for the technical support and the supply of necessary infrastructure.

Lastly, a big thanks goes to my loved ones, family and friends, who showed great support during the time of this thesis. They stood behind me and accepted my choice of priorities when it came right down to it and time was needed to work on this project.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As the Internet has grown over the years, professional demands grew stronger because more and more business is done over the Internet. Companies require their website or service being delivered to anywhere in the world—quickly and dependably.

To match these demands, various content delivery network companies have become operational with Akamai[1], Limelight[2] and Cloudflare[3] only being a few of them. These networks are used to deliver large amounts of data into every region in the world using caching and other, more sophisticated, technologies.

Some of the CDNs also provide accelerated delivery of non cachable contents, for example individual personalized pages for online shop systems etc. They try to achieve a performance advantage by splitting the transmission path into segments. To this end, traffic is sent to the Web server through multiple of their own servers so that some of the segments can then be rerouted to avoid known congestion nodes. This route optimization can only happen through a large overlay network, which the servers of the CDN form, and can be seen in the example illustrated in figure 3.1.

This thesis focuses on measuring the potential benefits of route optimization through overlay network routing and different routes will be analyzed for differences.

1.1 Research Questions

There are many aspects that can be explored regarding overlay networks. Using measuring methods for latency and other metrics, the thesis tries to answer the following research questions:

[1] `http://akamai.com`
[2] `http://limelight.com`
[3] `http://cloudflare.com`

How does optimized routing within an overlay network affect the latency of small sized data units?

Taking content editors working on centrally hosted real-time publishing platforms as an example. They edit documents using technologies such as *Operational Transformations* which are sensitive to latency and network failure, because every action gets sent to the server or the participants as an *operation* that transforms the old state into new state. Ultimately, this means that a lot of small packets are sent across the Internet on a frequent basis [1].

What impact does overlay network accelerated routing have on frequent but small packets when compared to common Internet routing using BGP?

How much jitter is observable on both transmissions?

How does optimized routing within an overlay network affect the download time of medium sized objects over HTTP?

Customers surfing the world wide web require individualized web site contents from centrally hosted shopping systems. When this content is used on an important page like the store's front page, the visitors should get fast results in order to be kept engaged.

What impact does overlay network acceleration have on medium sized packets around 100 kB?

Is there a benefit from optimizing the overlay network's TCP parameters?

How relevant is overlay routing when compared to other acceleration factors like transparent compression within the overlay network?

How similar are the transport routes that benefit the most from Overlay-Network-Acceleration?

When tracing overlay network and regular Internet routes, how similar are they?

How much difference can be observed in actual geographic location of intermediate hops, and how does the choice of network operator affect performance criteria such as round trip times?

1.2 Structure

This chapter just described a high level overview over the topic this thesis is about and poses research questions to be answered.

Chapter 2 provides background information by explaining terms used in this work and also lists related pieces of work and what differences there are when compared to this thesis.

While chapter 3 specifies the requirements that are necessary to do measurements that yield reliable results, chapter 4 will describe the implementation chosen to answer the questions from chapter 1. Since there were tools implemented to perform the measurements, these are explained in detail.

Chapter 5 focuses on the actual results and discuss various aspects found while doing measurements.

Finally, chapter 6 answers the research questions by summarizing the results found in chapter 5.

Chapter 2

Background & Related Work

2.1 Background

For a better understanding of this thesis, this section describes the most relevant concepts that are frequently used throughout the following chapters.

2.1.1 Internet

Autonomous System

While the Internet is commonly perceived as a single entity, it is actually composed of thousands of networks of different sizes. These networks can be of different types, like for example conventional **Internet Service Providers (ISPs)** that provide access to end users or content providers, or transit networks that exist to connect other networks.

Networks under a common administration are also called *Autonomous Systems* or short *AS*. Autonomous Systems don't only differ in the properties described above, but also in the way they are able to handle traffic, which is what Peering is for.

Peering

As outlined in the previous section, users and content servers oftentimes are not part of the same network. Therefore, the transit points between the autonomous networks play an important role for ISPs trying to provide a good service to their customers.

When two, possibly commercial, networks are connected and agree to not charge the other company for incoming traffic, the process is called *peering*. Peering is only meant to exchange traffic regarding the ISP's own customers, though—a network will never

route incoming traffic from a connection the ISP pays for (i.e. the upstream connection) to one of his peering partners.

When an ISP is able to reach every host on the Internet without paying for traffic, the network operator is a so called *tier 1* provider. Smaller ISPs are typically tier 2 or 3 providers that peer with other ISPs from the same tier. They are required to purchase upstream bandwidth from tier 1 providers or so called upstream providers in order to be fully connected to the Internet.

Peering negotiations, though important for the Internet's performance and health, can be of difficult nature when company policies are in conflict and the parties disagree on who is allowed to charge fees.

These negotiations exist at least since the late 1990s and are explained in detail in [2].

Upstream

While peering is one method of exchanging data between coequal networks, smaller ISPs with limited connectivity are required to pay for exchanging traffic with bigger networks.

These larger networks act as *Upstream Providers* and usually provide access to the whole Internet. As mentioned before, purchasing upstream bandwidth is the only way for non-tier-1 providers to achieve full connectivity to the whole Internet.

For better performing and more reliable connectivity, autonomous systems can be *multihomed*. That means that the ISP either uses multiple IP addresses on the same link or even separate independent links for his upstream connection.

Internet traffic is generally assumed to obey the concept of *valley free routing*, which means that a network provider will not use its paid upstream connection to accept or transmit packets originating from or destined for one of its peering partners. This is an important assumption for algorithms extracting knowledge about the Internet's topology from data flows and routing information.

Problems of Best Effort Routing on the Internet

No money in the middle mile is one possible name for problem that may prevent packets from taking the ideal path across multiple networks. This problem typically results from the economical aspects of inter-network peering.

When building Internet infrastructure, companies can sell data links to end users (last-mile) or to data centers (first-mile) which follow the rules of free markets. Faster con-

nections earn the ISP more money because there is an advantage over the competition, and the same holds true for data center up-link connections.

And indeed, performance of these links is improving steadily, as reports like [3] show. However, peering quality in terms of bandwidth, latency and availability are more or less hidden from the customers because while peering to one network may be excellent, other networks may be hard to reach and even encounter packet loss because of router overload.

Because of the peering agreements mentioned earlier, networks are not necessarily optimized to use the fastest paths for routing packets, but to be cost-efficient while still providing the service level that was sold to its customers.

BGP

The Border Gateway Protocol is an inter-domain routing protocol which means that autonomous systems operators exchange information about what part of the IP address space is reachable over a given router.[1]

BGP is a *path vector protocol* which means that routers include the complete paths when exchanging their own reachability information. This solves the count-to-infinity problem which is a common problem in distance vector protocols.

However, not all reachability information will be advertised by a router. *Policies* defined by a network's operator can prioritize certain routes and exclude other routes from being published. Otherwise, a network could announce its upstream connection to the peering partners which would be very costly and influence the networks stability because of a potentially higher load. On the other side, operators could choose to accept routes for the entire Internet only from its upstream providers but not from peering partners in order to avoid misconfiguration.

Nevertheless, a complex BGP setup can introduce serious outages or massively inefficient routing willingly or accidentally. Detecting routing problems could therefore lead to an improved stability and performance, as promised by some commercial overlay network providers.

IP

The Internet Protocol (IP) defines a rather simple mechanism for addressing hosts using globally unique addresses.[2]

[1] BGP Version 4: RFC 4271 – `https://tools.ietf.org/html/rfc4271`
[2] RFC 791 – `https://tools.ietf.org/html/rfc791`

The commonly used IP version 4 features an address length of 32 bits and therefore allows up to $4.29 * 10^9$ globally unique addresses available to address hosts. In reality, this number however is smaller because there are certain reserved IP address ranges that can not be used for addressing hosts. Additionally, companies from the early days of the Internet managed to each reserve a big chunk of the IP address space. This is because address space was handed out in blocks of $2^{24} \approx 16.77 * 10^6$ addresses (*Class A* IP address ranges) before *Classless Inter-Domain Routing* was introduced in 1993, which made allocating IP address space more efficient.

Because of the growing number of Internet users and the limited IP address space, IPv6 presented a simply enormous address space of 2^128 possible addresses.

This thesis will focus on IP version 4, and hence IPv4 addresses. However, nodes connecting via IPv6 are not expected to behave any different and will therefore not be analyzed separately.

TCP

The Transmission Control Protocol (TCP) is well known and used for most communication in the Internet. Working one layer above the Internet Protocol, it enables a stream-oriented communication between hosts over packet switched networks, such as IP networks.[3]

It also introduces mechanisms for multiplexing packet switched connections using port numbers, and allows the retransmission of lost segments and therefore enables reliable connections where packet loss is detected and corrected.

Finally, the protocol tries to avoid congestion at points within the network and at the receiver by effectively adapting the sender's data rate when packet loss is detected. The parameters for this behavior can be varied for different use-cases and network scenarios.

The downside of using TCP on a connection is the reduced performance because of two factors, which make opening a new TCP connection a quiet costly operation in terms of total transmission time:

1. Establishing a TCP connection requires exchanging 3 packets before the first real data can be transmitted[4]

2. The amount of data to be sent without waiting for a confirmation (congestion window) starts at a very small number and only then increases with the number of packets sent over the connection as long as the network supports the throughput

[3] RFC 793 – `https://tools.ietf.org/html/rfc793`

[4] TCP Fast Open, IETF draft at time of writing but already implemented in recent versions of the Linux kernel, accelerates this process

HTTP

Based on TCP/IP, HTTP (HypterText Transfer Protocol) has long become a standard protocol for exchanging hypertext content on the Internet.[5]

HTTP itself is a rather simple and generally stateless protocol that is extensible via custom headers. There are only few requirements when implementing a working subset of HTTP that works for most purposes, even though sometimes not being feature complete as per specification. This circumstance enabled its role as a transport protocol for not only hypertext but also media downloads or interactive communication. For example, AJAX is used for sending data in JavaScript Object Notation asynchronously within a web browser, even from low-profile hardware and embedded systems using HTTP.

In practice, there certainly are more complex features to be implemented for a HTTP server such as being able to serve the requested content type and encoding, respect caching requirements and more. Still, software like the Apache HTTP Server or NGINX among many others fulfill these requirements and are wildly available.

Keep-Alive, a feature of HTTP/1.1, is an important step towards an improved browsing or application experience, since it enables sharing of TCP connections for multiple HTTP requests. Using this feature saves round-trip times and enables higher throughput because of already *warmed up* connections.

DNS

The Domain Name System is used to resolve domain names like `spectre.net.in.tum.de` to an IP address like `131.159.14.85`.[6]

The system can be seen as a large distributed database where domain names can be resolved step by step for each label. Starting from the end of a domain name, a hierarchical tree of so called *name servers* is being traversed until a server that can answer the specific request is found.

Each name server is authoritative for his so called *zone*, which contains information about immediate sub-domains and the servers that are authoritative for those sub-domains.

The iterative process of query evaluation is usually not done by the end user itself, but is given to a *resolver* server instead. DNS resolvers have better connectivity to the Internet than end users and used by many people so that responses can be cached as long as the authoritative name server allows doing so.

[5] RFC 2616 – `https://tools.ietf.org/html/rfc2616`
[6] RFC 1034 – `https://tools.ietf.org/html/rfc1034`

2.1.2 Overlay Networks

When studying literature regarding overlay networks, one will find different meanings for the term.

Peer-to-Peer Networks

One of the prevalent uses of *Overlay Networks* is their notion as a network for peer-to-peer file exchange services like *Kazaa* and *Gnutella* [4, section 3.2, p. 3].

These networks tend to be highly dynamic and usually only consist of the end-user machines that are on-line momentarily. They are not purposed for the improvement of a given service but establish a completely new service themselves.

For the individual clients, their respective network and performance characteristics are not very important when participating in the network, because the goal is not to achieve great performance and low download times, but to distribute a large files at a very low cost by leveraging the participation of every user in the network. Also, distributing files across many users drastically improves availability and reduces single points of failure.

Using only the users' home Internet up-link connection still enables reasonable download rate for other users without renting infrastructure like servers.

Content Delivery Networks

Another popular area of overlay networks are content delivery networks, which are intended to improve a given existing web service by delivering data to a large amount of users.

They achieve content acceleration by various techniques, such as caching and compression of static assets on a server that is close to the customer. This leverages lower round trip times and more reliable connections so web sites load faster and companies can distribute much greater amounts of data, because the actual delivery is spread widely across the Internet.

This meaning can be found in [5, p. 8] where the Akamai Network is described as an "high-performance overlay network" multiple times.

In contrast to the average Peer-to-Peer network, these networks are highly organized with well performing hardware located at strategically chosen, highly interconnected data centers.

While technically dealing with the same problem as more dynamic networks, like dysfunctional or temporarily unavailable hosts, the networks' nodes are more stable and participation changes less than with personal end-users.

Other

Because there are many other kinds of overlay networks, giving a comprehensive listing is neither possible nor intended for this work. However, a few more types shall now be described briefly.

- **TOR** (The Onion Routing)[7] is a project featuring an overlay network providing anonymity for its users. This is achieved by routing packets through multiple intermediate hops (*relay nodes*) before reaching the actual target server within the TOR network. But not only TOR internal servers are reachable without being noticed by e.g. totalitarian regimes. Servers on the public Internet are contacted using TOR *exit nodes* after which traffic is yet again observable in the clear.

  Anonymity is provided by wrapping the original packet in multiple layers of encrypted routing information. Only one layer can be decrypted by each TOR node so that the ultimate target location is obfuscated for all intermediate nodes. TOR is, however, susceptible for attacks via traffic analysis by collecting usage data over a longer period of time [6]. Safe and anonymous communication for an individual is therefore not absolutely guaranteed.

- **VPN** (Virtual Private Network) software is available from many commercial and open source maintainers. VPNs provide an encrypted connection between two computer systems or networks and behave like a physically existing connection to the network stack's higher layers. The newly created virtual private network then forms a logically decoupled overlay network over public Internet infrastructure.

- **6in4**[8] is one of the many technologies used to master the migration from IPv4 to IPv6. A host that has only IPv4 connectivity, but wants to use the IPv6 protocol, can append that IPv6 packet to an ordinary IPv4 header with its `protocol` field set to $41_{(10)}$. It can then be sent through a preconfigured tunnel while only adding 20 Bytes of additional data because of the IPv4 header.

- **RON** (Resilient Overlay Network) is an application layer architecture built by distributed servers on the Internet that measure interference and disconnects between each other. It aims at detecting outages faster than the currently used routing protocols and therefore providing a more stable network for distributed applications.

  The Massachusetts Institute of Technology's RON project is one well-known implementation of a resilient overlay network and its project page[9] features many resources like research papers and actual measurement data.

[7] `https://www.torproject.org`
[8] RFC 4213 – `https://tools.ietf.org/html/rfc4213`
[9] `http://nms.lcs.mit.edu/ron/`

2.1.3 Akamai Web Performance Solutions

Akamai offers a variety of services to improve companies' Internet presences.

For example, Akamai offers cloud storage systems and web application firewalls that reduce the load on a company's web server and protect it from threats like hacking attempts and denial of service attacks. Next to that, there are services that accelerate website delivery and downloads by combining multiple methods using transport- and content optimization.

The Akamai *Edge Network* consists of 170,000 servers in 102 countries within over 1,300 networks [7], but these numbers have grown further since.

Content Delivery Network

To deliver content to a large number of users, Akamai *Globalhost (GHost)* Servers, which is the technical name for servers in the *Akamai Edge Network*, perform load balancing, caching and video live streaming.

For download services and delivery of static data, a caching hierarchy is used in order to grant fast transmissions and avoid passing requests through to the origin server in times of high demand.

For video live streaming, a technology emulating IP multicast is used to keep traffic close to the origin to the minimum and replicate the data as late as possible on the route to the client [5].

Akamai Overlay Network

The servers of the Akamai Edge Network form an Overlay Network over the Internet which is fully interconnected using public Internet infrastructure only.

Even though any node can potentially speak to any other given node, only certain connections with good performance between nodes with high network capabilities are used.

TCP Optimization

At Akamai, TCP is used for GHost-to-client, GHost-to-GHost, GHost-to-origin, GHost-to-netstorage communication and therefore is an essential subject for optimization. This is why TCP parameters regarding the congestion window and retransmission times, among others, are modified [8].

Even though TCP connection handling is usually part of the operating system's kernel, the optimizations can be used in different settings for each customer. The chosen degree of optimization for non-cachable content depends on the service level agreed on.

For example, a highly optimized configuration can be optimized to set an initial congestion window size so that a typical packet can be transmitted without waiting for acknowledgments. Also, TCP's slow start phase can be avoided for a new connection if there currently are connections in use, because values like the congestion window are remembered to benefit additional connections.

The Edge Server configuration allows the configuration of almost every TCP parameter regarding waiting times, retransmission protocols and the handling of inactive connections. However, remembering and modifying the congestion window as mentioned above is probably the most important factor for standard TCP optimization.

Persistent TCP Connections

As described before, TCP is important for communication between GHost servers. For improved performance, Akamai uses persistent TCP connections between GHost servers. These connections are established on demand and can time out when not used after a certain amount of time.

Without persistent TCP connections, any new communication flow between two points in the overlay network would first have to perform the TCP three-way handshake. With persistent connections, there is a good chance of a connection already being available and therefore no overhead is generated.

As opposed to regular Internet routing where packets are forwarded directly, multi-hop overlay routing would be bound to deliver a worse performance—simply because of all the handshakes needed for each segment in a route.

SureRoute

SureRoute is a service available for higher customized and more expensive Akamai contracts. It enables the acceleration of non-cachable contents by influencing the transportation route between client and origin server. While still using connections of the public Internet, packets are sent from one Akamai Edge Server to another until there is no improvement possible anymore and the origin server is contacted.

To enable the SureRoute service, there is a set of highly interconnected nodes steadily measuring connection performance between each other. Congestion points or link failure on the Internet can thereby be identified quickly using frequent probing.

By intentionally forwarding packets to another Akamai Edge Server, these congestion points are avoided and performance is expected to be better than with conventional routing via BGP [9]. This is expected to hold true even though usually more IP hops than on the conventional route are traversed.

To determine the best path to the origin server (i.e. the next Akamai Edge server to contact), SureRoute races are started by requesting the *SureRoute Test Object* at two preselected parent nodes and the origin server. For these races, the response time is measured and the result is cached for the next requests.

Because waiting for the initial SureRoute result should be avoided, the first request always hits the origin server directly. SureRoute acceleration will then be used for the following requests afterwards, but is not bound to always contact another Akamai Edge Server. If no optimized route can be found, the Edge Server can simply forward the request to the origin server and wait for the reply, which will also use an overlay accelerated route back to the client.

2.1.4 PlanetLab

PlanetLab is a scientific computer network to test and deploy new network services or applications in a large scale environment.

Since the start of the project in 2003, more than 1000 researches have tested applications for distributed storage, network mapping, peer-to-peer systems, distributed hash tables and more. At the time of writing, PlanetLab consists of 1353 nodes at 717 sites.[10]

While a large portion of nodes are located in Europe and North America, fewer nodes also exist in Australia, New Zealand, Asia, South America and even Africa.

Creating an account requires participating in the project by providing two nodes oneself. After that, PlanetLab offers shell access on computers around the world so that own software can be installed and run.

2.1.5 Masurement Tools

ping

Generally available in all GNU/Linux-, BSD-, and Windows-based computer systems, the *ping* program sends *ICMP* (Internet Control Message Protocol) *ECHO_REQUEST* packets to a given network host and measures the time it takes to receive the response packet. There is no retry mechanism and therefore a lost packet is no uncommon event.

[10] `https://www.planet-lab.org/`

The tool is used to measure the RTT (Round Trip Time) between the current computer and another host, given that no intermediate router or firewall will drop the packet because of filled buffers or ICMP-avoiding filter rules.

cURL

cURL is a popular tool used to perform HTTP requests from the command line.

It is is widely available and allows not only the exact configuration of passed parameters for the HTTP call, but also provides a compact syntax to specify an output format of request meta-data like elapsed time and number bytes downloaded.

traceroute

traceroute is another fundamental network administration and Internet measurement tool. It gathers the IP addresses of IP hops between the executing host and a target host by provoking ICMP TTL-exceeded errors for each hop. The response IP addresses are recorded and displayed in order so that the route can be understood.

Traceroute is available on most platforms as `traceroute` or as more advanced programs that essentially feature the same functionality like `mtr` and `pathping`.

2.2 Related Work

2.2.1 Overlay Networks

In [10], the authors compare the effects of best-path probe-based reactive overlay routing in contrast to redundant multi-path routing.

The former is an approach that is very closely related to the technical background behind Akamai's SureRoute service where periodic probes are used to gain information about the current state of routes. In the paper, this approach is primarily focused on the detection on packet loss or host and link failures. Because probing overhead is seen as a big problem of this routing approach, timing methods are explained to reduce the traffic generated by probing nodes.

Another approach on reducing the probing overhead of overlay networks is described in [11] where algebraic methods are used to find an upper bound of monitoring $\mathcal{O}(n \log n)$ linearly independent routes being sufficient for a network consisting of n nodes. Also, [12] focuses on this problem and tries to reduce the number of connections that need to be monitored for packet loss while still getting a good overview over the current network conditions.

It is also pointed out in [10] that frequent probing and the thereby implied overhead may be unbearable for lightweight data streams, but definitely justifiable for large data streams. The efficiency depends on the kind of constraint experienced in the network and can therefore vary between *no benefit* and *great benefit*. It is emphasized that routes with a low loss rate may not also feature low latency and the other way around.

The second method for improving transmission reliability presented involves sending packets over both the regular Internet and, additionally, over a randomly chosen node that acts as a relay station. The authors explored that choosing this method reduces the chance of losing two packets sent over an intermediate nodes only by 10 % to a still high chance of 60 % when compared to sending the two packets back-to-back on the same path with a chance of losing of 70 %.

Choosing the correct strategic positions for nodes in an overlay network is described as an NP-hard problem in [13], but it is also shown that around 100 nodes are enough to enable routing over shortest paths from a single source into all autonomous systems. Also, the average path length is reduced by 40 %, which is a very important figure for TCP optimization and use cases demanding very low latency.

2.2.2 Latency Measurement

Since latency measurements are an important part of this thesis, related work in this field can enhance the workflow and the result's consistency and significance.

The very recently published [14] presents a method for latency measurements on a very large scale. It uses statistical methods to collect latency data very efficiently computation and memory wise. However, the proposed *COLATE* system is designed to also be implemented in the network itself and focuses on the storage efficiency. For this work, a simpler approach can be taken that relies on classical timestamping but can be evaluated without sophisticated statistical analysis.

Because of its target to measure HTTP transaction latency, [15] is an interesting article on passive measurements of the time-to-first-byte in situations where active probing cannot be used. The authors present various metrics and estimations based on values that are easier to measure, like the round trip time. Round trip time measurements in this theses have the advantage that active probing is not only possible but in fact the only source of traffic that can be observed, since both web server and clients used for this work serve no real-world purpose. Also, the article's focus lies on the comparison of different connection technologies for mobile users like UMTS, LTE & co., and not on the comparison of different transportation routes.

2.2.3 Route Similarity & Comparison

Paxon et. al. discuss end-to-end Internet route properties like stability, symmetry and also similarity in [16], which is highly relevant for this thesis as well. In order to compare the similarity of two routes, a normalized version of the Levenshtein Edit Distance is used. That means that the Levensthein distance is divided by the length of the longer route being compared, because it would otherwise positively correlate with the routes' lengths.

The article studies the effect of load balancers on route diversity and also compares different hop identifiers like country, autonomous system, city, address prefix and IP hop and compares these metrics between 2006 and 2009.

While featuring impressive accuracy and profound results, the focus is on changing metrics over a long period on time. For the comparison between overlay network routes and plain Internet routes, the article achieves to give interesting ideas, especially regarding the variety of choices for the hop identification.

The *iPlane Nano* system described in [17] is representing enough of the Internet's topology in a database of 7 MB with an update stream of approximately 1 MB per day. Using only this data, it is capable of predicting 70 % of AS paths exactly, estimate latency within 20 % for over 60 % of paths and also predict loss rates very accurate for over 80 % of paths. It can be used to implement Internet scale peer-to-peer applications that need to estimate routes between participants in order to exchange data efficiently.

The article also goes into depth on how to model the complex process that Internet routing is. While the proposed *iPlane Nano* surely works great under the assumption of *valley free routing* under normal conditions, it is probably rather useless for predicting routes between the same points when it comes to overlay routing, as it would be needed for this thesis.

Chapter 3

Design

For answering the questions posed in the beginning of this work, this chapter describes which steps are necessary in order to perform measurements revealing the effects of overlay routing on a higher level.

3.1 Overlay Routing

This section will explain the most basic concept of why overlay routing is expected to yield performance benefits compared to regular Internet routing. To do so, figure 3.1 shows a very high level schematic view of some connections through the Internet.

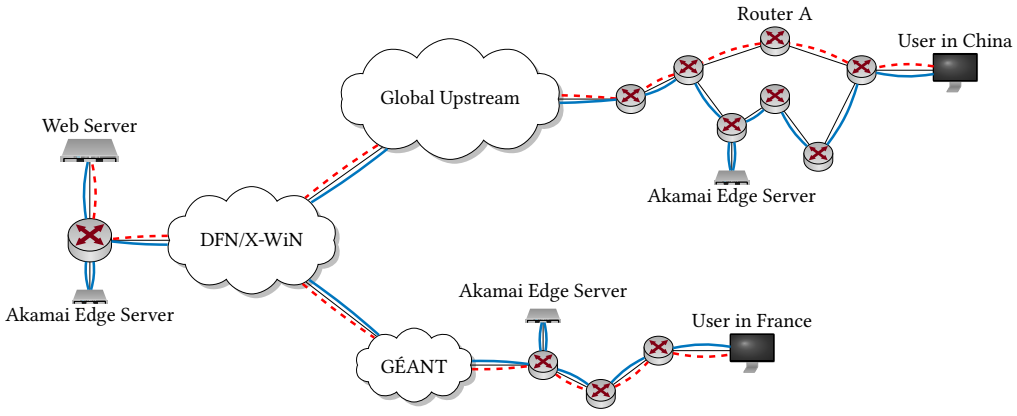


Figure 3.1: Example of diverging transport ways when using Akamai Overlay Network routing and plain Internet routing

On the left side of the figure, there is a web server which acts as the central part in delivering content to the test nodes throughout this thesis. The web server is not only reachable over a global upstream connection, but is also connected to the European

scientific network *GÉANT*. Using this scientific network, many universities in Europe and the whole world can be reached using a high bandwidth infrastructure.

A hypothetical user in a location far away from central Europe, for example a user in China, would connect to the web server using the upstream provider and many intermediate IP hops, simply because of the long geographic distance. On this route, there would be a lot of intermediate routers so that, when transmitting data to the web server, there is a realistic chance of traversing a router that is congested because of high load. In the figure, this would be *Router A* which is used by the red dashed line that represents regular Internet routing using BGP (see 2.1.1).

An overlay network, for example the one instituted by Akamai, can gain an overview of Internet congestion and packet loss rates by constantly monitoring connections between strategically placed servers. When found that the transport way that uses *Router A* is slower than the best known route, explicit hop-to-hop routing between the Akamai servers can be used to actively avoid congested network routes. The blue continuous line represents the way data takes when routed through the Akamai Overlay Network.

Note that the blue Akamai route is almost certain to be longer than the original route (if not because of different intermediate routers then because of the hops introduced by the Akamai servers themselves). However, it can still be faster than the original route because of reduced latency and packet loss since congested routers are being avoided. This is the expected case for long transport routes with many IP hops, for example with clients in countries that are far away and connected using the public Internet only. Even when the figure displays the Akamai route to go through an Akamai server immediately before reaching the web server, this would not be necessary and Akamai servers on the right could also directly contact the web server themselves.

As seen in the lower branch, it's also possible that the transport ways for Internet and Akamai don't differ at all, if the route found using BGP is already very good. This is the expected case for clients residing at academic institutions that are connected to the web server using an optimal connection already.

3.2 Answering the Research Questions

How does optimized routing within an overlay network affect the latency of small sized data units?

The first research question asks for overlay routing latency benefits when compared to the regular Internet routes. Generally spoken, the question can be answered by collecting and analyzing round trip time measurements over both transport ways.

What makes measuring RTTs harder in this scenarios is, that Akamai severs only act

as very sophisticated HTTP proxies (as opposed to IP relays or VPN servers), so layer 3 round trip times can't be measured for the whole way through the overlay network. Instead, only the path between the client an the first Akamai server can be measured using ordinary tools such as `ping`.

The only way to measure end-to-end round trip time is by measuring the duration of small sized HTTP requests sent over an already established TCP connection. By using payload sizes small enough to fit into one packet per request, round trip times that closely approximate the underlying layer 3 latency can be measured.

How does optimized routing within an overlay network affect the download time of medium sized objects over HTTP?

For the second question, a similar approach can be taken.

Once more, completion times of HTTP requests have to be compared for both routes to the origin web server. Instead of transmitting payload data that fits into one packet, a larger payload should be used to simulate a real download.

How similar are the transport routes that benefit the most from Overlay-Network-Acceleration?

This research question can be answered by comparing both the regular Internet and the overlay network route on a large scale. As some of the papers in the second chapter's *Related Work* explained, a distance metric such as the Levensthein Edit Distance can be used to compare routes by their IP hops, AS hops and other criteria.

Although anticipating implementation details of this work, it can be said that it is only possible to trace routes through the overlay network by using manual lookups, which cannot be automated. Still, by combining `traceroute` measurements with the manual lookups, it is possible to do a certain number of case studies where characteristic routes can be examined at a hop-to-hop level.

3.3 General Requirements

In order to gain solid knowledge about the effects of overlay network routing, there are several demands regarding the measurement setup and execution.

Diversified Measurement Locations

The potential performance benefits from overlay routing has to be measured from as many different locations as possible to gain a good overall picture and receive reliable average values of the differences to regular BGP routing.

When evaluating the data, it may be important to select a group of nodes sharing certain properties, i.e. a long path to the target server or a certain geographic location. However, this selection has to be possible after collecting the data, so it's important that the experiment isn't restricted to a certain set of nodes unnecessarily.

The computers made available as servers which can be used for experimentation are usually much better interconnected than the devices of typical home or mobile users. This has to be kept in mind when working with commercial overlay networks that in general aim to improve the average user's Internet experience.

Sufficient Measurement Period

Since the Internet's performance strongly depends on time-dependent traffic characteristics, measurements have to be taken over a longer period of time spanning a few days, a whole week, or even longer.

Capturing a whole week of measurement data potentially enables recognizing usage patterns based on time of day and day of the week. This prevents coming to a conclusion that is only true for a certain point of time.

Simultaneous Start

When using a large number of test nodes, the measurement setup has to be able to connect to all of them as simultaneously as possible in order to facilitate the comparability of measurement data.

Larger differences in the starting time of continuous measurements would result in increasingly different network conditions which would influence the performance that was being measured.

Fault Tolerance

Since nodes are expected to be unavailable from time to time, the setup has to be resilient against unavailable nodes and failing executions.

The experimentation progress must not be stopped by a failing node while data integrity has to be maintained. That means that the experimentation tool must not crash when

trying to contact many nodes at the same time with some of them failing to respond. On the other side, it also mustn't deliver wrong or outdated results when newer ones would be missing.

3.4 Technical Requirements

3.4.1 Web Server

A server running a standard web server application that can serve the required amount of page hits—both bandwidth- and CPU-wise.

Number of Requests per Second

The average number of requests should be estimated in advance by roughly following the formula:

$$\text{Requests per Second} = \frac{2 * \text{Number of Measurement Nodes}}{\text{Interval Between Requests in Seconds}} \tag{3.1}$$

The *number of measurement nodes* is multiplied by 2 because both request durations—the default path's through the Internet and the overlay network routed path's—should be measured at the same time.

The number of requests per seconds is reduced by an interval duration which defines the number of seconds to wait between sending new requests.

Outgoing Bandwidth

For download tests that generate a mentionable amount of outgoing traffic at the origin server, its uplink capacities should be known in advance.

Similar to the formula above, the average outgoing bandwidth can be calculated according to:

$$\text{Outgoing Bandwidth} = \frac{2 * \text{Number of Measurement Nodes} * \text{Object Size}}{\text{Interval Between Requests in Seconds}} \tag{3.2}$$

The factor 2 is, again, because of the comparison of the two different transport routes, overlay network routed and default Internet routing.

Object Size could be a typical downloadable object from the World Wide Web and measure around 100 kB. However, for very small objects and many test nodes, the

transmission overhead created by IP, TCP and HTTP should be taken into account.

A quick back-of-the-envelope calculation with around 16 Bytes on Ethernet based systems, at least 20 Bytes for the IP(v4) header, 20 Bytes for the TCP header and additional 30 Bytes for HTTP requests or 200 Bytes for HTTP response headers. Adding those numbers up leads to about 70 additional Bytes for requests and 240 Bytes for responses.

The *Interval Between Requests in Seconds* again describes the duration between requests. For larger download payloads it would typically be set to a larger duration than for smaller latency tests.

Admittedly, these calculations are only true under the assumption that the incoming requests are distributed evenly and don't come in bursts. However, this assumption should hold true because of different latencies and slightly varying starting times for the nodes of the experiment.

Hosting

While the server's location is not the highest priority, uplink diversity can play an important role when measuring the advantages of overlay network optimized routing.

High end data centers usually have multiple uplink providers for increased availability. An overlay network routing system has to detect link failure and switch to another link if possible. However, these link failures would typically occur in the middle mile of the transport route and the overlay work should then be able to reroute traffic accordingly. Enabling multiple completely independent routes starting from the data center can still be an important criterion for enterprise grade services.

When benchmarking the performance of overlay networks, it may therefore be interesting to look at the ability to direct requests to the origin server over a different uplink once failure on the original link is detected.

3.4.2 Software

Design

In order to make the results of this thesis as reproducible as possible, as many openly available tools as possible should be used.

If possible, these tools should have been used in network engineering and administration for a long time to make sure they produce reliable outputs and are well understood by a wide audience.

Still, own software has to be implemented when the available tools lack support for features needed to investigate the questions posed in this thesis. When writing new software, it has to be kept in mind that all code written should be easy to understand for verifying the results of this thesis. This also enables reuse in potential future research projects and includes documenting the code and choosing a permissive open source license.

File Format

Measurement data should be collected in a format that not only contains the raw time values, but also metadata regarding the time of measurement, the nodes' IP address and the interval between requests.

For analysis, these documents may be converted to a more suitable format after ensuring data integrity and plausibility.

Chapter 4

Implementation

This chapter describes what precise steps where taken to gain the information described later in the thesis and follows the requirements described in chapter 3.

4.1 Server Setup

The first steps towards the experimentation setup is setting up a web server used to deliver the HTTP objects when doing round trip time measurements. This section lists in detail which software has been used and what configuration has been altered.

4.1.1 Webserver

For delivering the HTTP objects used for the measurements, a virtual server running Debian GNU/Linux 8.0 with one multi-threading CPU core running at 2.67 GHz and 1 GB RAM equipped was used. Swapping was enabled but not necessary due to the low memory requirements of the installed software.

The web server software was Nginx[1] in version 1.6.1 serving the default `index.html` or binary files that were generated at the beginning of the experimentation setup using data from `/dev/urandom`. Nginx was preferred over the Apache HTTP Server because the former is known for performing particularly well in simple configurations for delivering static files. However, the decision between any commonly used HTTP servers should hardly matter with today's hardware capabilities[2].

The server has gzip compression disabled in order to deliver the same amount of bytes regardless of the requesting client's capabilities to avoid accidentally distorting mea-

[1] `http://nginx.com`

[2] Using NGINX as the web server produced less than 5 % CPU load while serving around 300 requests per second during early tests

surement data. Also, the clients will include no information regarding the HTTP cache control mechanism, which means that every request gets served the same way.

For HTTP-Pings, persistent TCP connections (HTTP keep-alive header) is used. The server is configured to deliver up to 1 000 000 requests and will wait up to one hour of inactivity between requests before closing the connection.

The number of worker threads have been limited to 1 with a maximum connection limit of 4096 connections per worker. This was done to provide a linear performance impact when the number of test nodes increases. Otherwise, more active test nodes could result in a better average performance because of more workers serving requests simultaneously, which means that workers would be less busy each and therefore respond to requests faster.

4.1.2 Hosting

While the web server is physically located with a direct connection to DE-CIX Frankfurt, it is logically part of the chair's network, including the IP address. The actual upstream providers are the Leibnitz-Rechenzentrum (LRZ) which is connected to the Deutsches Forschungsnetz (DFN/X-WiN) which is one of the providers peering at DE-CIX and which is connected to other upstream providers in order to reach out to the whole Internet.

Because of this setup, data will always be routed through the chair's infrastructure before being sent back to the DE-CIX using a transparent tunneling mechanism. However, these connections are stable and powerful enough so that the difference between the logical and physical location of the server only adds a static offset to all the round-trip time measurements.

Due to having the DFN as a direct upstream provider, there is an excellent connection to other academic networks in Europe and Northern America, where many of the Planetlab test nodes are located.

The physical up-link bandwidth of 200 Mbit/s is enough to sophisticate the requirements, the server's fully qualified domain name is `spectre.net.in.tum.de`.

4.1.3 Akamai SureRoute

For the comparison between regular routing and Akamai's implementation overlay network routing implementation, a customer account has to be created, which thankfully was enabled through the company's cooperation.

Configuration

Keeping in mind that the routing benefits should be the only difference to be measured, any other possible optimizations have to be turned off.

In particular, this required changing the configuration to reflect this state:

- Disabled TCP optimization
- Disabled caching
- Disabled (potentially transparent for the end-user) compression (within the Akamai network) via gzip

As required by Akamai, a typical web site with a size of 33KB is used as the SureRoute test object when Akamai servers perform their races determining the best route.

The web server's SureRoute accelerated version was reachable under `spectre-akamai.net.in.tum.de`.

Usually, HTTP keep-alive sessions are closed after a given total time, a maximum time of inactivity or a maximum number of requests. This is done to prevent too many open connections that block resources on the web server or intermediate proxy servers.

While Akamai Edge servers support persistent connection for multiple HTTP requests over one TCP connection, the connection will be terminated after answering 2000 requests, as early tests showed.

DNS Setup

In order for Akamai to serve requests from their Edge Network instead of from the origin server, the end-user has to be pointed to a suitable Edge Server, which is done using DNS.

When a customer starts using Akamai services to accelerate its web contents, he changes the public DNS name for his site to not point to the origin server directly anymore, as it would be the case with a common website setup.

Instead, it points to a customer specific Akamai domain. For example

```
spectre-akamai.net.in.tum.de
```

does not resolve to the origin server's IP address, but is instead a `CNAME` entry pointing to

```
spectre-akamai.net.in.tum.de.edgesuite.net
```

which is within the authority of the Akamai name-server (`edgesuite.net` is a domain owned by Akamai).

Real world customers would of course not suffix the domain name with an Akamai specific label, but instead introduce a new domain for internal use that points directly to the origin server. This work uses `-akamai` to clearly differentiate the accelerated server from the regular web server.

The Akamai DNS server is now responsible for resolving the intermediate domain name (ending in `edgesuite.net`) to an actual Akamai Edge server that can serve the origin's content.

Naturally, choosing an Edge Server is a crucial step that relies on a sophisticated decision process considering not only the end-users distance to the Edge Server, but also the availability of already cached contents for the given origin [5].

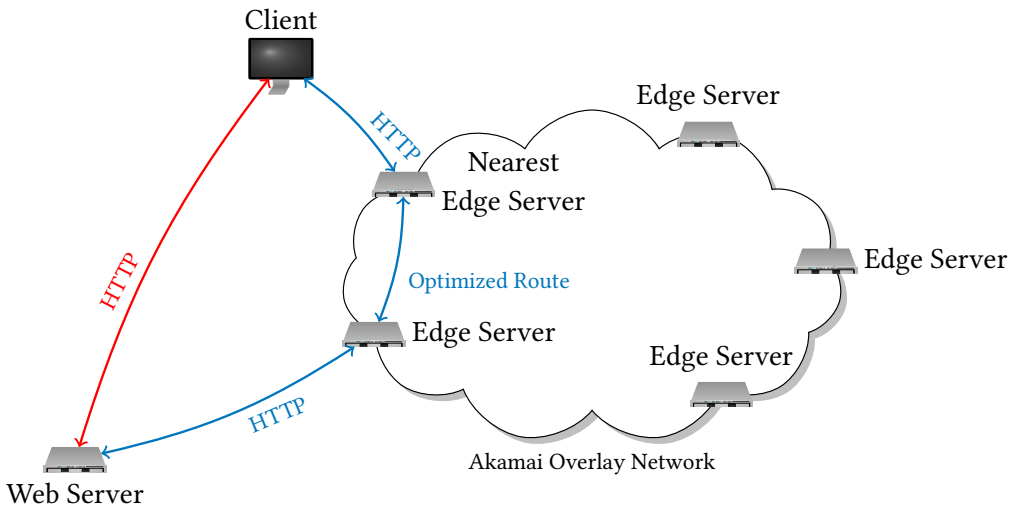


Figure 4.1: Illustration of using the nearest Akamai Edge Server for overlay network routing

Figure 4.1 shows why this complex DNS setup is required. Only when using the Edge Server which is closest to the client, finding an optimized route through the Akamai overlay network can bring an advantage over simply connecting to the web server directly. This works because the domain that is intended for end users (in this setup `spectre-akamai.net.in.tum.de`) is actually a `CNAME` chain resolving to to the closest Akamai Edge Server which is then used as an entry point into the overlay network. After contacting it, the requests gets passed to either another Edge Server or the origin web server.

4.2 Measurement Tools

4.2.1 `htping`

The section Answering the Research Questions in chapter 3 already explained why performing ICMP-style pings over persistent HTTP connections are required to measure the performance benefit. The tool `htping` was implemented to fulfill this task.

Features

`htping` supports measuring against multiple target hosts and will send the specified number of requests to all hosts synchronized at the same time using the given time interval to wait between sending.

The round trip time is recorded using precision timers that are based on the systems steady clock and are not affected by fluctuating CPU clock rates etc..

To support evaluation based on time slices, the start time of a measurement is saved to the output file. Because nodes may differ largely in their system time, the time-stamp is requested using a HTTP based web service. The expected difference of up to one second because of network and HTTP processing latency is within a passable range and much less than without this external synchronization.

Additionally, the public IP of the measurement node is recorded and saved, so test nodes can be filtered or grouped by their location afterwards.

`htping` is implemented in the Rust[3] programming language, which provides low level access and performance to ensure correct time measurement and access to TCP connection with minimal overhead.

Different to compilers for C or C++, the Rust compiler has the advantage of offering a greater level of memory safety and program correctness in concurrent environments.

Options

Table 4.1 shows all options options are available and are printed when the program is invoked with `--help`.

The program also expects to find at least one hostname to be measured as in the example invocation in listing 4.2.

Furthermore, it will refuse to run when neither `--out` nor `--verbose` are given to prevent faulty operation by the experimentator.

[3] http://rust-lang.org

Option	Shorthand	Default Value	Description
`--count`	`-c`	10	Number of requests to make per target
`--interval`	`-i`	1000	Send request with this interval in ms
`--out`	`-o`		Write measured data to file at path
`--verbose`	`-v`		Be more verbose
`--help`	`-h`		Print this table

Table 4.1: `htping` options

When `htping` receives a `SIGINT` signal, it will stop sending requests, save the measurement data collected until the signal occurred and shut down. This feature can be used when an earlier measurement should be replaced by a freshly started one during the experiment.

Program Flow

A typical execution flow:

1. The program starts and determines the test node's public IP and fetches the current time from an external webservice.
2. The program sequentially initializes a TCP connection to both hosts and waits until they are established.
3. For each connection, it starts a receiver thread that
   (a) reads from the respective connection
   (b) saves the time-stamp when a response was received
4. Then, it starts a sender thread that, as long as the target count has not been reached and both connections are alive:
   (a) sends a HTTP request on each connection, save the current timestamp
   (b) calculates duration until the $(n+1)^{\text{th}}$ request should be sent by comparing the current time with the target time of (absolute_start+$(n+1)*$interval_length)
   (c) sleeps for that duration
5. Finally, the program substracts the *received-response timestamps* generated in (3b) and the *sent-request timestamps* from (4a) to get the duration for each request.
6. It generates and saves the output file.

Measurement File Format

For postprocessing, a plain text format is useful because it is easier to integrate with common Unix/GNU tools such as `gnuplot` or text processing tools.

However, measurement data collected by `htping` contains metadata including the start time, the test nodes public IP address and the used interval between requests.

JSON[4] was chosen as the measurement file format because of its flexibility, lightweight syntax and availability in every environment.

For each given origin, the time needed to establish the TCP connection is stored next to the IP address the hostname resolved to. While that is the same for the origin host over plain BGP routing, the IP address will vary greatly when using the Akamai hostname.

The data array contains the raw time values of each successive request in milliseconds, but with a resolution on the nanosecond level.

Listing 4.1: Example `htping` JSON file

```
{
  "timestamp": "2015-05-07T21:32:53Z",
  "public_ip": "80.165.237.10",
  "interval": 5000,
  "results": [
    {
      "host": "spectre.net.in.tum.de",
      "target_ip": "131.159.14.85",
      "time_connect": 170.641152,
      "data": [
        110.585038,
        163.588215,
        // ...
      ]
    },
    {
      "host": "spectre-akamai.net.in.tum.de",
      "target_ip": "80.165.238.9",
      "time_connect": 191.029369,
      "data": [
        238.465119,
        160.055152,
        // ...
      ]
    }
  ]
}
```

Example

For this thesis, this invocation was used multiple times:

[4] `http://json.org/`

Listing 4.2: Example `htping` invocation

```
$ ./htping spectre.net.in.tum.de spectre-akamai.net.in.tum.de \
    --verbose --count=2000 --interval=5000 --out=htping.json  \
    > htping.log &
```

This invocation includes the following specifics:

- Measure the hosts `spectre.net.in.tum.de` and `spectre-akamai.net.in.tum.de`
- Perform 2000 requests and exit afterwards
- Wait 5000 ms between requests
- Save the structured measurement data to `htping.json`
- Save the whole output containing status messages and possibly errors to `htping.log`
- Start the program in the background so the measurement orchestration software (gplmt) can terminate before waiting until all requests have been made

Because of the 2000 requests per connection limit on Akamai Edge servers, the `htping` is restarted every 2 hours (less than the run time of $2000 * 5s = 2.77h$) to avoid gaps between the continuous measurements of the different runs.

4.2.2 Measurement Instrumentation

While not directly relevant for the measurements' results, the process of instrumenting the measurement process itself turned out to be more complex than originally estimated. As described below, network usage limitations and software incapabilities required a major change of the toolchain from an existing project towards custom made shell scripts. The individual components are now being described in detail.

GPLMT

The GNUnet Parallel Large-scale Management Tool (GPLMT) is a tool to deploy and run experiments remotely on a large number of systems in parallel.[5]

To orchestrate the experiment and instruct the test nodes to setup and run the measurements, GPLMT uses a XML description of the commands to execute on the nodes over SSH. A collection of commands is called a tasklist.

In addition to executing tasks, GPLMT supports transmitting data from and to the test nodes using SCP or SFTP. This can be used to retrieve results after the tasklist is completed.

[5] GPLMT user guide

What makes GPLMT specifically useful for this work is its ability to retrieve the list of nodes directly from the PlanetLab API and run a taskfile on around 1000 nodes within a few minutes when using maximum parallelism.[6]

Because of the Keep-Alive limit on Akamai hosts, the experiment has to be restarted about every two hours, which also yields the benefit of activating hosts that were temporarily unreachable at an earlier try.

SSH Connections

Because of the large number of nodes, a large number of SSH connections have to be instantiated every time a new round of measurements is started. Also, these connections should be created within a small amount of time, because the measurements on all nodes should all cover the same period of time. Delaying connections would mean varying measurement periods which results in more difficult post-processing.

Coming to a conclusion, these requirements mean the creation of 1000 SSH connections within a few minutes, every two hours.

Even though the experiment instrumentation server was white-listed at the university's Internet provider, this behavior resulted in automatically triggered abuse e-mails to the chairs network administrator and associated instances. While not critical, this circumstance is highly unwanted because it shows the chair in a bad light and should be avoided. As a side effect, using persistent SSH connections as described in the next section also enable more accurate measurement instrumentation because of the faster access to the PlanetLab nodes.

SSH ControlMaster

In order to solve the problem of frequently instantiating a great number of SSH connections, SSH offers a mechanism called *ControlMaster*. ControlMaster enables the sharing of multiple sessions over a single network connection.[7] Setting the option to *auto* instructs SSH to use an existing socket file for communication to the host but will create a new connection if necessary.

Unfortunately, as GPLMT is written in Python, it uses the Python library *paramiko*[8], which does not recognize the ControlMaster settings in the SSH config file. Since working through the source code of GPLMT/paramiko and fixing the problem there would have been a complex task and not much functionality of GPLMT was needed

[6] Given that the ISP allows the creation of that many SSH connections within a short period of time, which is typically not the case with home Internet connections

[7] `man ssh_config`

[8] `http://www.paramiko.org/`

anyway, the decision was made to use custom shell scripts to do the measurement instrumentation.

`sshed`

Using ControlMaster does not per se solve all the requirements—SSH connections still need a few seconds each to be established. Therefore, running an experiment would still cause the creation of a lot of connections simultaneously.

Instead, a new bash script is introduced, `sshed`. Its whole purpose is the maintenance of the ControlMaster connections. The script accepts a list of hosts and tries to bring up as many connections as possible within the given interval of e.g. one minute.

In order to avoid bursts of connections, the actual connection time for each host is randomized within the interval window. Once a connection is established, it is kept alive by sending a simple `echo` command. For unreachable hosts, however, the script would try to build an new SSH connection in every interval, which would result in an even higher number of connection attempts than the original approach using GPLMT. Since the number of unreachable hosts is expected to be even greater that the number of working hosts, this poses a real problem.

Borrowed from established network systems such as Ethernet[9], a *binary exponential backoff* mechanism is implemented. This means calculating a probability P for every host h which in each interval is used to decide whether to attempt connecting to the host or not:

$$\Pr[\text{“visit host } h\text{”}] = \frac{1}{\min\{2^n, 1024\}} \tag{4.1}$$

In this formula, n is the number of unsuccessful connection attempts which gets reset when the connection attempt succeeded and that is initialized to `0` to produce a 100 % chance of checking a host once when in the first interval. If the number of hosts is really large, n can also be initialized to e.g. `2` in order to reduce the number of connections when first starting `sshed`. This, however, increases the time until the connection pool is *saturated* and experimentation can start.

As can be seen in the formula, the smallest possible probability is $\frac{1}{1024}$, which would result in checking a maximal inactive host every $\approx$ 8.5 hours on average, given an interval time of one minute.

In addition to the `sshed` script, the SSH config is set up to detect failing connections within the time slot of 60 seconds by using the `ServerAliveInterval 60` option for all hosts.

[9] IEEE 802.3

Command Templating

Now that maintaining SSH connections are taken care of, a simple script can be used to repeatedly execute the measurement command on every host and start the measurement segments.

When re-running a command after some time, result files from an earlier run must not be overridden. As it turns out, there is no particularly simple and compatible way to automatically write to files with incrementing filenames on a given Linux system.

That means that one can't simply run a command

```
ping spectre.net.in.tum.de > ping.log
```

every two hours, because the result would get overwritten. The desired output would be generating files such as `ping_00.log`, `ping_01.log` etc., but as said above, this is hardly possible.

Instead, a simple substitution step is implemented that converts an command template file to the actual command file by replacing a defined placeholder such as `{{n}}` with the number of the current measurement.[10]

Hereby, a *command template file* is nothing else than a shell script containing a set of placeholders, such as `{{n}}` which will get replaced by real values in order to retrieve the actual command file.

Listing 4.3: Example command template file before substitution

```
# run htping on origin
./htping spectre.net.in.tum.de spectre-akamai.net.in.tum.de \
  --count=2000 --interval=5000 \
  --out=htping_{{n}}.json -v > htping_{{n}}.log &

# ping origin
ping -c 2000 -i 5 -s 800 spectre.net.in.tum.de \
  > ping-origin_{{n}}.log &

# ping akamai
ping -c 2000 -i 5 -s 800 spectre-akamai.net.in.tum.de \
  > ping-akamai_{{n}}.log &
```

Listing 4.4: Resulting command file with run number 42 inserted

```
# run htping on origin
./htping spectre.net.in.tum.de spectre-akamai.net.in.tum.de \
  --count=2000 --interval=5000 \
  --out=htping_42.json -v > htping_42.log &

# ping origin
ping -c 2000 -i 5 -s 800 spectre.net.in.tum.de \
  > ping-origin_42.log &
```

[10] This can be achieved using the *sed - stream editor for filtering and transforming text* command

```
# ping akamai
ping -c 2000 -i 5 -s 800 spectre-akamai.net.in.tum.de \
  > ping-akamai_42.log &
```

While this process seems complicated at first, it is very easy to implement and efficiently solves the problem of consecutively numbering result files, as in the example above.

Of course, the parameters could have passed in to the script when executing, but for compatibility reasons, creating the final script with all variables already filled in was preferred.

Command Execution

Taking the processed command file from above, it now has to be run on all the hosts that are currently reachable. This is done using a rather simple SSH script that, every two hours, asynchronously runs a function for every host, that:

1. Checks whether the SSH connection for the host is currently available
2. Runs the processed command file by passing its content as an argument to SSH

4.3 Post-Processing

In order to utilize the data generated in the measurement steps for knowledge discovery, post processing is a crucial factor.

4.3.1 Data Collection

Collecting the data is completely decoupled from experiment instrumentation, even though it works very similarly. For starting measurements, a script will iterate over all hosts and check whether a SSH connection currently exists and if that is the case, a command will be issued.

For result collection, `scp` is used because it can be instructed to use the SSH ControlMaster feature and therefore work very fast on existing connection. Since the data being transferred is usually plain or structured text, compression can speed up the process dramatically. `scp` can be told to use compression via the `-C` flag.

In practice, measurement data segments can be collected at any point, even while an experiment is still running. This allows early-on insights revealing problems that would otherwise have gone unnoticed until the final data exploration.

4.3.2 Data Processing

After collecting the data as described above, the large amount of request times has to be processed in order to gain insights into the different transport routes.

Processing Pipeline

To do the different steps in post-processing, a processing pipeline has been implemented in Python. The core of it is a very simple script that just loads a JSON configuration file, loads the tasks referenced and runs each tasks with the previous task's output as an argument.

Additionally, every task can have an own options hash where specific settings matched to the relevant data set can be specified. This design enables the reuse of the processing pipeline in future projects because tasks can be swapped as needed and the base is easy to understand.

Processing Steps

To only give a short overview over the steps taken to produce the graphs that can be seen in chapter 5, this is a high level explanation of one processing pipeline:

1. **Glob** for a specified file pattern belonging to the current data set in order to create a nested list of directory names (hosts) and file names (measurement segments).
2. **Concat** all JSON files belonging to one node, producing a list of tuples containing the response time measured for both transport routes and an absolute timestamp.
3. Create **time slices** from the list of measurement times by selecting exactly one tuple from each node for each time slice of the measurement's sampling duration. A time slice then contains one list for both transport ways containing the RTTs from each node.
4. Calculate **statistics** such as the median, first quartile, third quartile etc. for each time slice.
5. **Plot** the time slices' statistical values in the appropriate way, outputting a `.pdf` graphic for embedding in this thesis.

When comparing multiple node groups using box plots, nodes can be preselected before the first step or filtered by top level domain.

4.3.3 Visualization

Data visualization is an important step in the process of knowledge discovery. Choosing the appropriate visualization technique can lead to the discovery of correlations and support the creation of hypotheses. On the other side, graphics can also be misleading and make the viewer recognize properties that aren't representative for the rest of the data.

For generating graphs, the Python library `pyplot` is used which is part of `matplotlib`[11]. Its dependency, `numpy`[12] is a package that brings many basics for scientific computing in Python such as data structures and algebraic functions.

Scatter Plots

Scatter plots are a good starting point when exploring a dataset because it can show points on a two dimensional plane and even compare a low number of different data sets by using color coding or different symbols for each set.

For this work, the x-axes in scatter plots are always used for indicating the time that passed since the start of the experiment. The y-axis on the other hand displays the round trip times that were measured for the requests at a given time.

Generating scatter plots can give a good overview over the data contained in a set because one can easily see how much points are spread or condensed at any point in time. When comparing different sets in one figure, it is also possible to tell whether both sets are affected by an irregularity or not.

Unfortunately, scatter plots become hard to read when a larger amount of data is shown. As soon as points starts overlapping each other, very dense areas tend to become one continuously filled area. It is then impossible to estimate a difference between two regions within such areas.

Also, when comparing different data sets with similar values, they typically overlap and later sets may be drawn over other sets. This problem increases with the number of data sets to compare and even using transparency and blending to create new colors where points overlap becomes confusing quickly.

Regression Lines

An obvious solution to the problem of overloading a graph is to reduce the number of data points being displayed by combining values in a meaningful way. When only the

[11] `http://matplotlib.org/`
[12] `http://www.numpy.org/`

overall trend matters, for example when the overall performance of both routes should be compared, its favorable to use a *linear regression function*.

A linear regression line is a polynomial with a degree of 0 that approximates a set of points as good as possible. In practice, that means that the calculated line minimizes the squared error by comparing the interpolated point $p(x_j)$ with the actual point y_i for every point (x_j, y_j) in the data set of length k. The error itself is calculated according to the formula

$$E = \sum_{j=0}^{k} |p(x_j) - y_i|^2 \tag{4.2}$$

Of course the polynomial function can be of any degree to interpolate a corresponding signal, but for analyzing a trend, grade 0 is the appropriate choice.

Box Plots

When drawing a regression line combining a large amount of data segments, one has to choose which value to represent per x-value. This can be for example the median, first quartile, third quartile, minimum value or others.

In order to represent as much information as possible while not loosing overview because of the growing number of lines, box plots can be used to represent several aspects of a data set on the y-axis while using only one unit on the x-axis.

Typically, a box will be drawn that has its bottom line on the first quartile and its top line on the third quartile. In between, the median is marked by another horizontal line.

Above and below the box there are the whiskers that mark the 5^{th} and 95^{th} percentile[13]. These are typically connected to the box using a dashed line that is horizontally centered.

Box plots are great to compare different selections of a data set because they offer a lot of information while using limited space.

[13] Other works may use whiskers to display the 1^{st} and 99^{th} percentiles

Chapter 5

Evaluation

This chapter evaluates the results collected in the experiments regarding the different research questions. Additionally, a short evaluation of the PlanetLab network featuring some statistics collected for this thesis is given.

5.1 Experimentation Setup

Over the course of this thesis, PlanetLab was a substantial part for collecting measurement data as it provided computers that were accessed in an automated way using SSH. Unfortunately, it also brought up multiple problems which had to be solved in order to perform scientifically accurate measurements. Of course, some of the concerns are referable solely to the large amount of node, like the large number of SSH connections that conflict with the university's network policy.

PlanetLab itself is divided into two projects that are heavily connected. Nodes in Europe are organized under the PlanetLab Europe (PLE)[1] project while organizations outside of Europe belong to PlanetLab Central (PLC)[2]. However, members of any of the two projects can also use all the other testbed's nodes as well and organize them over one single web interface.

5.1.1 Node Availability

A rather surprising fact when testing PlanetLab functionality was the portion of nodes that was actually available.

[1] `https://www.planet-lab.eu/`
[2] `https://www.planet-lab.org/`

Of the 1048 nodes that were added to the slice using PlanetLab's web-based control interface, at most 311 were reachable at the time these statistics were created, which equals around 29.7 %.

A short analysis on the reason why a large majority of nodes was unavailable was done and is represented in table 5.1 (PlanetLab Node Availability Summary).[3]

Amount	Status	Comment
311	Working	
414	Timeout	Establishing the SSH connection timed out
140	Not Found	Resolving the node's hostname failed
84	Authentication Failed	The SSH key was not accepted by the node
71	Connection Refused	The node or its firewall refused the connection
19	Unreachable	The node or its network was not reachable
9	SSH Error	The node's SSH server produced an error
1048	total	

Table 5.1: PlanetLab Node Availability Summary

This evaluation shows the importance of a fault tolerant measurement instrumentation setup. However, also the post processing process needs to deal with missing measurement segments for certain nodes because of temporal unavailability that collided with starting a new measurement segment.

5.1.2 System Versions

The nodes' system versions are an important criterion for choosing available tools or even developing own software. The following data is extracted from the 311 available nodes but the numbers don't add up to that amount because as with every experimentation made, some nodes may have been unavailable at the time of measurement or executing the command simply failed.

Kernel Version

Using the `uname -r` command prints the operating system kernel's release. Table 5.2 (PlanetLab Node Kernel Versions) shows the raw data when counting occurrences of the different releases.

As it can be seen, 284 of the 292 nodes (97.3 %) are running on the Linux kernel in version 2.6.32 or higher. Even though version 4.1 has been released by now, this is a solid foundation for a large scale facility such as PlanetLab to run on and should not interfere with working with modern tools.

[3] Failure comments are deduced from status codes and may in parts represent a temporal condition

Count	Kernel Release
158	2.6.32-20.planetlab.i686
79	2.6.32-36.onelab.x86_64
28	2.6.32-36.onelab.i686
12	3.11.10-100.fc18.x86_64
8	2.6.27.57-33.planetlab
5	2.6.32-34.planetlab.x86_64
2	2.6.32-131.vs230.web10027.xidmask.2.mlab.i686

Table 5.2: PlanetLab Node Kernel Versions

The kernel release string also yields some information over the systems architecture. 96 of 292 nodes (32.9 %) are running a 64-bit system while the vast majority of 67.1 % are running a 32-bit system. It it therefore absolutely necessary to compile own software in a manner that is compatible with both architectures, i.e. that runs on 32-bit systems.

Operating System Versions

Next to the kernel versions, the operating system plays an at least equally important role because it defines which versions of common software is available from the repositories without having to compile it from source.

Table 5.3 (PlanetLab Node Operating System Versions) shows the two versions of Fedora Linux[4] that are used on PlanetLab nodes. While it is positive for the developer that there are only two versions in use, the operating systems are rather old. According to the list of Fedora releases and historical schedules[5], version 8 (Werewolf) was released around 8 years ago in November of 2007 and version 14 (Laughlin) only three years later on November 2nd 2010. This makes it more difficult to work with up to date software. On some systems installing new software even fails because of outdated SSL certificates and requires extra work for enabling the use of software repositories without SSL.

Count	Fedora Version	Codename
174	8	Werewolf
112	14	Laughlin

Table 5.3: PlanetLab Node Operating System Versions

5.1.3 Node Distribution

While not necessarily being relevant for every experiment that happens on PlanetLab, the geographic location of test nodes plays an important role when performing latency

[4] `https://getfedora.org/`

[5] `https://fedoraproject.org/wiki/Releases/HistoricalSchedules`

measurements against a single location. Since the later is the case for this thesis, the node locations are inferred from the nodes' DNS top level domain part and shown in table 5.4 (PlanetLab Node Distribution by Country).

Also, a rough differentiation into PlanetLab Europe (PLE) and PlanetLab Central (PLC) is made, although there are some nodes differing in its umbrella project even though the TLD is clearly European or non-European. The bottom entries with only one or two nodes per top level domain are combined into one row each for the sake of brevity.

Count	Project	Estimated Location
94	PLC/PLE	USA[a]
16	PLE	France
16	PLE	Germany
16	PLC	China
14	PLE	Spain & Catalonia
12	PLE	Poland
11	PLC	`.com` domains, not country bound
11	PLE	United Kingdom & Ireland
10	PLC	Japan
9	PLE	Greece
8	PLC/PLE	`.org` domains, not country bound
7	PLE	Portugal
7	PLC/PLE	`.net` domains, not country bound
7	PLC	Canada
6	PLC	New Zealand
6	PLE	Italy
6	PLE	Finland
6	PLE	Czech Republic
6	PLE	Belgium
5	PLC	Argentina
4	PLC	Brazil
3	PLC	Singapore
3	PLE	Norway
3	PLC	Hong Kong
7×2	PLC/PLE	Thailand, Sweden, Russia, Mexico, South Korea, Israel & Australia
8×1	PLC/PLE	Slovenia, Romania, Netherlands, Hungary, Ecuador, Denmark, Cyprus & Switzerland

Table 5.4: PlanetLab Node Distribution by Country

[a] Possibly including institutions in other countries with `.edu` domains registered before 2001

It can be seen that 103 of 308 nodes (33.4 %) are located on the continent of North America, 120 nodes (38.9 %) in Europe and 26 nodes (8.4 %) in Asia, so these continents account for at least 80 % of available nodes. However, the share is likely to be higher

because of the 26 nodes (8.4 %) that can not be located precisely because the domain names belong to either the `.com`, `.org` or `.net` TLD. On the other side, there are two nodes in the `.fr` TLD that are actually located on *La Réunion* in the Indian Ocean.

These statistics were created in the beginning of this work in order to evaluate the expressiveness of measurements on the PlanetLab nodes. Just before starting the final series of measurement however, it seemed like PlanetLab Europe started to experience problems regarding the federated login mechanism that allows members of PLE to access nodes belonging to PlanetLab Central. As a result, only the nodes from table 5.4 that are marked with PLE were available for performing measurements.

5.2 Non-Accelerated Route Measurements

For the first experiment, the goal was to find out about the static overhead that using an overlay network from an outside client (i.e. a home computer) would inflict. That means, the amount of time it takes to contact the overlay network's nearest server and do the processing that is required to route the request—regardless of whether a possible acceleration would happen afterwards or not.

Because of the chair's internal infrastructure setup, arrangements have had to be made in order to keep the latency between the web server used during the experiments and the public Internet as steady as possible. Therefore, all round trip time measurements will show an additional overhead, originating from inside the chair's network.

Nevertheless, measurement accuracy when comparing transport ways or node groups is always guaranteed because each and every request was affected the same way.

5.2.1 Measurement Overview

To gain information about the typical delay the extra hop would cause, **HTTP HEAD** requests were sent over both the plain Internet and the overlay network's route. Measuring the response times then gives the pure RTTs[6] because requests were sent over an already established TCP connection and therefore avoided performing the typical handshake which would distort the results. For executing the experiment, the `htping` tool has been created from scratch and is described in section 4.2.1.

HEAD requests were chosen for this experiment because the overlay route's RTT benefit was to be explicitly excluded from the measurement values and Akamai does not accelerate HEAD requests using SureRoute. This way, the Akamai Ghost server simply takes the public Internet route as well and the overhead of contacting the overlay network's server first can be measured.

[6] except for a very short processing time needed by the web server

For figuring out that overhead baseline, 103 PlanetLab nodes performed HTTP HEAD requests every 5 seconds for 14 continuous hours for both routes. The round trip times were measured and stored in blocks of 2 hours each which results in 721 possible segments at maximum. In fact, 711 (98.6 %) of those segments actually completed successfully which results in the number of total requests being 2 047 680 for both transport routes combined.

5.2.2 Results

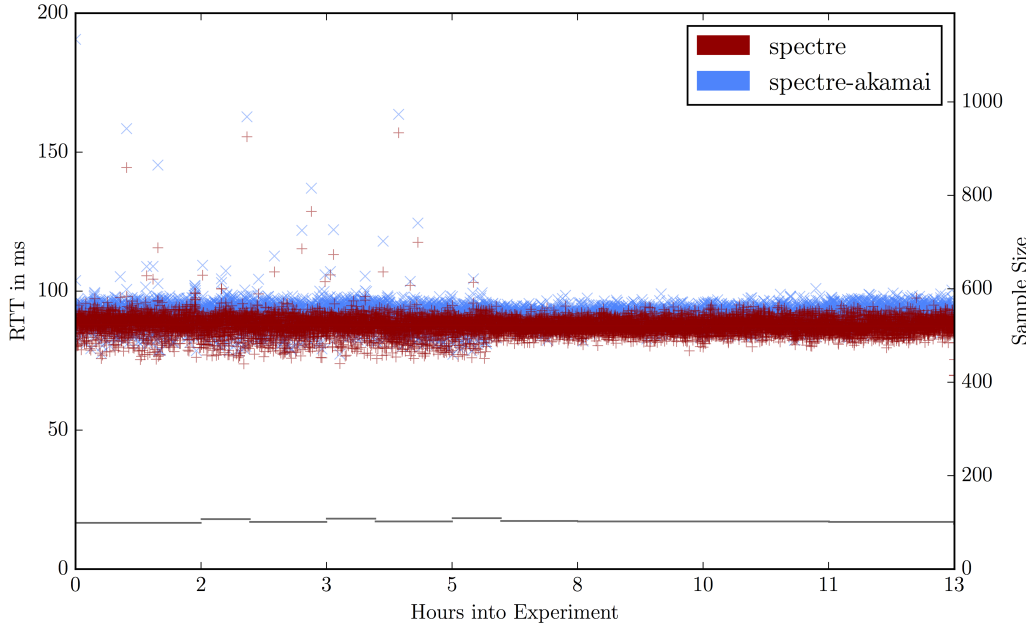


Figure 5.1: Scatter plot of `htping`-measured RTTs using HEAD requests

In figure 5.1, round trip times collected as described above are plotted with each data point marking one median value of all RTTs measured within a time slot of 5 seconds. Red *pluses* mark a RTT median value measured over the plain Internet route, blue *crosses* stand for a value measured over the Akamai accelerated route. The gray line at the bottom of the graph shows the amount of nodes that were measuring both routes at any point in time.

It is clear that many of the blue Akamai times are greater than most of the red regular Internet times which would mean that overlay routing does not provide any benefit here. Still, it is very hard to define the exact difference because plotting a large amount of data often results in misleading graphics where the process of drawing suggests wrong conclusions. Figure 5.1 features so many data points that it is impossible to gain further insights.

As this problem has already been described in 4.3.3, the logical consequence is to use linear regression function instead to gain more insight on the data. Creation of such a function is explained in section 4.3.3 and won't be discussed any further in this place.

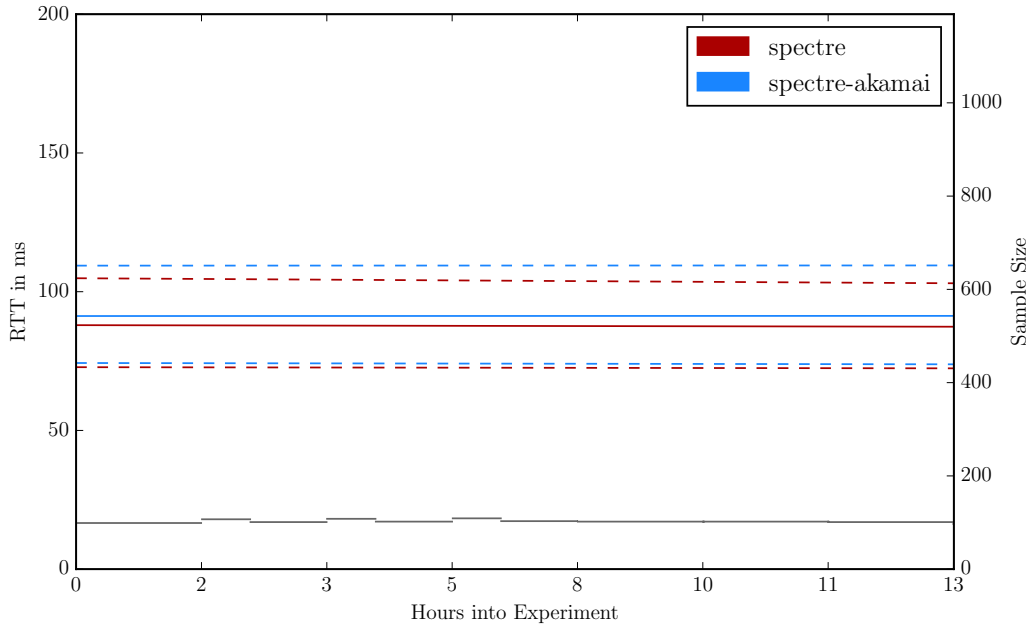


Figure 5.2: Linear regression plot of `htping`-measured RTTs using HEAD requests

After this modification, it is now possible to determine the maximal and minimal difference between the both routes' RTTs in figure 5.2.

Reading from the chart, requests taking the overlay network routed path are, in their median, between 3.3 ms and 4.0 ms slower than their counterparts over a regular Internet connection.

5.2.3 Conclusion

This overhead in round trip times that was measured while doing head requests seems rather unexpected at first, not only because it seems to invalidate the presupposition explained in section 3.1 (Overlay Routing), but also because it contradicts the propositions made by companies such as Akamai.

However, the intention of this experiment was to quantitatively measure exactly this initial overhead, which can be seen nicely in figure 5.2. When evaluating round trip times with acceleration, this baseline has to be kept in mind in order to assess the actual round trip time benefits.

5.3 Accelerated Route Measurements

To answer the first research question, *"How does optimized routing within an overlay network affect the latency of small sized data units?"*, round trip time measurements featuring HTTP requests to a web server have been made as described in the section 3.2.

5.3.1 Measurement Overview

For measuring round trip times over both regular Internet routing and the Akamai overlay network, a set very similar to the first experiment (Non-Accelerated Route Measurements). Again, `htping` is used to send **HTTP GET** requests over persistent connections to the web server and measures the response times.

Measurements were planned to be running over a period of at least one week so that it would have been possible to identify a difference in the measured values between working days and the weekend. Unfortunately, these measurements turned out to be more complex than originally thought. Not only the measurement instrumentation setup, which had to be switched from GPLMT to the custom SSH shell scripts, but also outages and planned maintenances at the university's infrastructure interfered with long-time measurements.

In the end, it was possible to collect data over 32 continuous measurement segments lasting 2 hours each which means that data has been collected over the course of 2.5 days. In total, 103 test nodes produced 2839 segment files equaling round 27.56 segments per node. If every node had been reachable all the time and no errors during the measurement had occurred, the total number of segments should have been $32 * 103 = 3296$ files. Compared to the actual number of successfully retrieved measurement segments, the measurement process showed a reliability of 86.13 %.

Of course, the amount of measurement segments alone is not that interesting. Extrapolating from the numbers stated above, 4 088 160 HTTP requests have been performed for each transport way. This number is calculated by multiplying the number of measurement segments with the number of requests made during each measurement period of two hours, which itself can be found by dividing the duration of 2 hours by the 5 seconds interval between each two requests.

Counting both the requests made over the unmodified Internet route and the Akamai accelerated route, the web server served 8 176 320 requests used for RTT measurement.

5.3.2 Results

To gain an overview over the collected data, figure 5.3 shows the scatter plot of each time slice's median value.

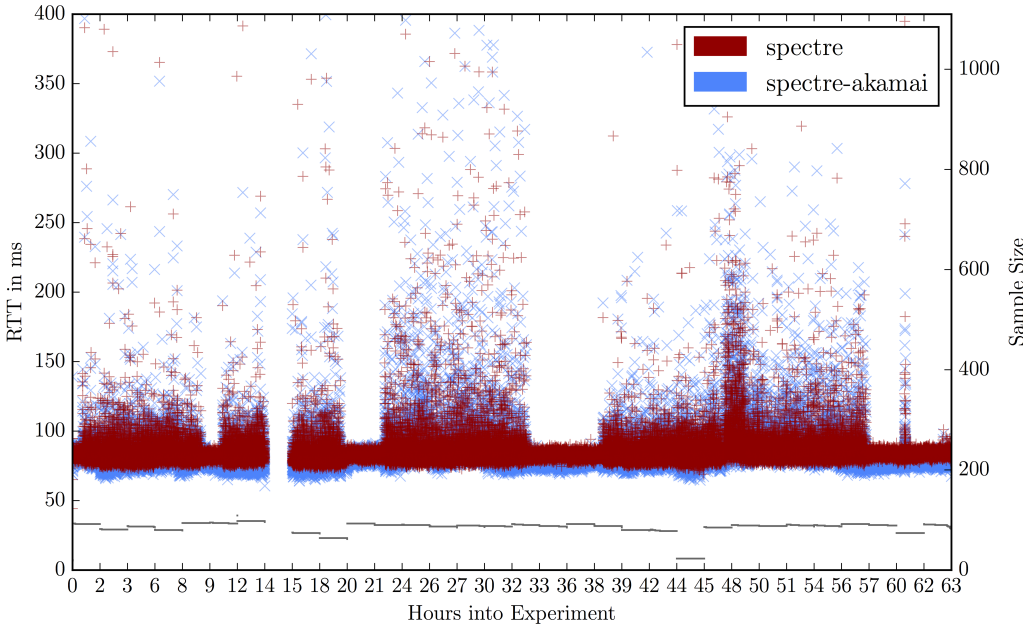


Figure 5.3: Scatter plot of `htping`-measured RTTs using GET requests

As can be seen, the amount of nodes that were taken into account while calculating the time slices' median values are fluctuating quiet a lot where at some point only 25 nodes delivered a measurement result. Most of the time however, around 90 points per time slice and route have been used which grants an overall reliable result. The gap between hours 14 and 15 denotes an actual gap in the measurements where less than 5 nodes were reachable and therefore left out of the evaluation because the results in that timespan would not be substantive.

The graphics also shows that there is an enormous variance in response time in certain periods of the experimentation. Concluding from the fact that the graph shows median values instead of average values, which would be more susceptible to outliers, it can conjectured that the problem causing the deviation affects all nodes in a similar matter. It is therefore most likely that there was an issue lying close to the web server, for example in the chair's network which gets passed-trough by every packet going to the web server.

As seen with the scatter plot of HEAD requests, no definite conclusions can be made from figure 5.3 as it is too crowded and imprecise. For this reason, another form of plotting the data has been chosen and is presented in figure 5.4.

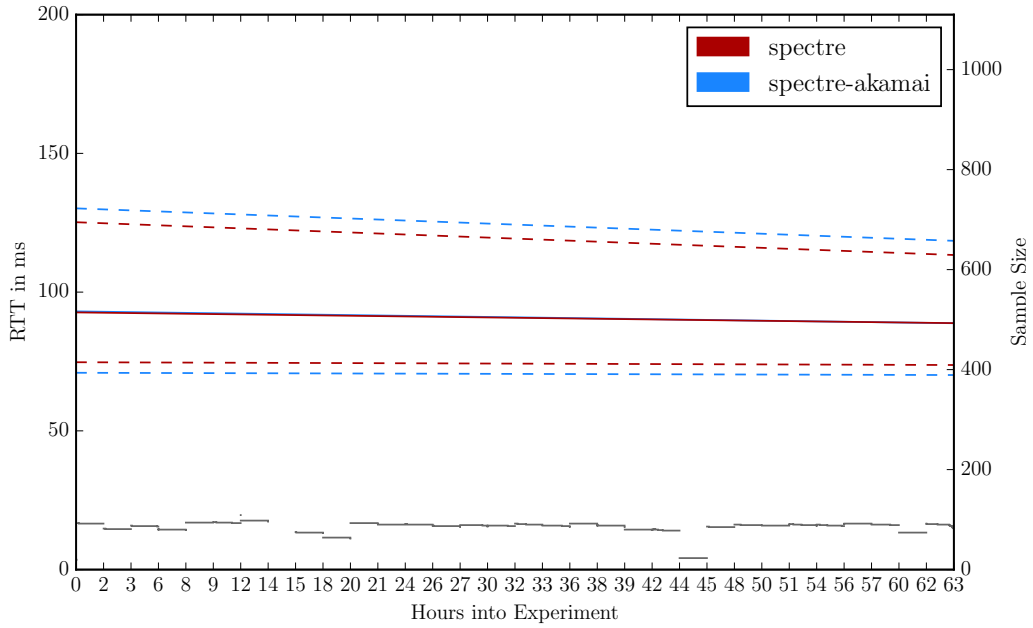


Figure 5.4: Linear regression plot of first quartile, median and second quartile RTTs measured by `htping` using GET requests

In addition to the both median regression lines shown in red and blue having nearly the same values, also the first and third quantile are shown by using dashed lines. The lower two lines represent the trend lines for plain Internet and Akamai 25th percentile, the upper two lines stand for the 75th percentile.

That means that when starting from the bottom of the graph, for each transport way, the first line marks the round trip time that is greater than 25 % of all the RTTs and the middle line the value that is greater than 50 % of all RTTs. Finally, 75 % of all round trip times of each transport route are lower than the topmost dashed line of the respective color.

Evaluating the Trend Lines

Comparing the trend lines of figure 5.4 reveals an interesting circumstance.

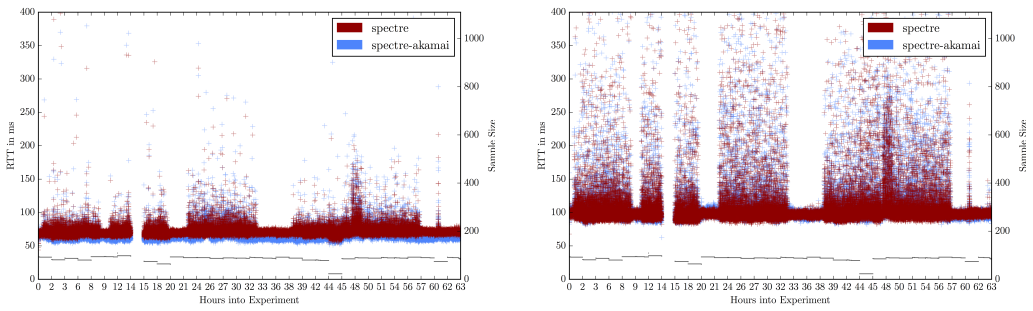
First of all, there is virtually no difference between the two median lines in the middle, which is not particularly surprising since the data consists of both kinds of nodes: The ones that are close to the web server which should perform best using regular Internet routing, but also the nodes farther away that should perform better using the optimized route. Therefore some nodes will profit by using overlay network routing by experiencing decreased round trip times and others will measure increased RTTs because of the overhead explained in 5.2 (Non-Accelerated Route Measurements).

However, neither the first quartile line nor the third quartile line conforms with the intuitive expectation that good routes with low response times should benefit less from overlay network acceleration than the routes with a higher response time. Instead, it may seem like already good RTTs are getting better when using the overlay network's route and rather bad RTTs become even worse.

As already mentioned, this is a highly unexpected result which demands an explanation.

To investigate this discrepancy, figure 5.5 shows a scatter plot of both 25th and 75th percentile RTTs for the same data set shown in 5.3 and 5.4. Obviously, the center of data points in figure 5.5a is below the center of 5.5b. The more interesting fact can quickly be seen is the different deviation shown by the two plots.

While the 25th percentile's points are fairly concentrated and don't show a high variance, the points on the right graph vary at lot with a strong drift towards times over 200 ms.



(a) 25th percentile

(b) 75th percentile

Figure 5.5: Scatter plot comparison of 25th and 75th HTTP GET percentile

Inspecting the left graph in figure 5.5, the overlay network route performs better than the standard Internet route when analyzing the time slices' low round trip times. Since this behavior is shown is when looking at the 25th percentile, that means examining the median of the lower half of measured values in a time slice. As seen in the previous regression lines, the overlay network actually perform better because it has more points below the plain Internet route, which is consistent with the graphic above.

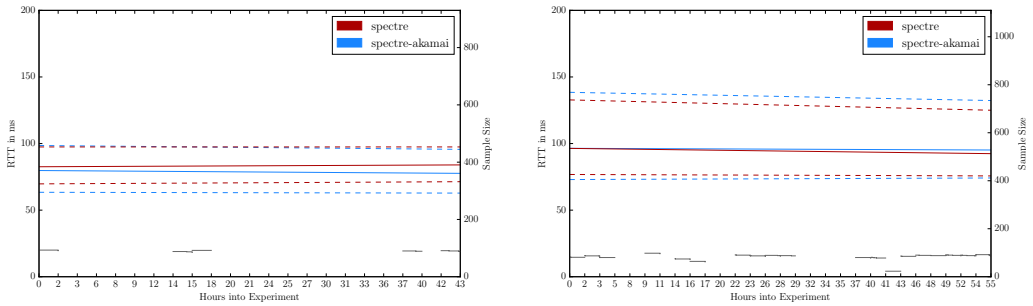
On the other side, the overlay network route performs worse than the standard Internet route when analyzing the time slices' high round trip times. In contrast to the 25th percentile, the 75th percentile is the median of the upper half of measures values, i.e. the slower performing requests. However, it cannot be said with certainty that the overlay network performs significantly worse in this sector, which means that the outliers must have greater values and therefore influence the overall quartile line.

On the Informative Value of Regression Lines

The previous section tried to reason about an easy to make misconception regarding the expressiveness of the regression lines shown in figure 5.4, but without leading to any reasonable explanation.

To avoid these misconceptions, it is very important to understand that the difference in first and third quartiles does not in any way mean that routes with already low RTTs are getting accelerated even more and routes with high RTTs get even longer. Instead, it shows that choosing the overlay network route only increases the deviation of round trip times when there is a source of jitter on the way, as it is the case for the respective time periods seen in figure 5.3.

To support that statement, figure 5.6 compares the regression lines for the same set of nodes but with 5.6a only showing values where no jitter was observable and 5.6b only uses data from periods with great influence.



(a) Only segments without jitter

(b) Only segments with jitter

Figure 5.6: Regression lines of all hosts but separated by network effects

In 5.6a it can be seen clearly that the lines match the expected outcome of overlay routing: Some node gain an advantage and therefore the round trip times in general are a little lower, which is why the blue 25^{th} percentile and median lines are a little lower than the corresponding red ones.

The fact that the overlay network's 75^{th} percentile is higher than the plain Internet one's in 5.6b is very interesting, because looking back at figure 5.4, it turns out that the overall regression lines look more similar to the regression lines in the right graph seen above.

Since there are around 3 times more segments experienced the network effects than the ones that don't, it is no surprise that calculating an average value over inhomogeneous data can be misleading when there really are separate cases to be compared individually.

Analysis by Geographic Location

An important aspect of analyzing the round trip time benefits overlay routing can bring is of course the comparison of different geographic locations. Table 5.4 (PlanetLab Node Distribution by Country) already gave an overview of how well countries are represented in the PlanetLab projects.

In order to find groups of nodes that should share a common distance to the web server, a `traceroute` measurement was run that yielded the path length between each of the test nodes and the web server measured in IP hops. However, these numbers were not as meaningful as expected, because nodes within one country varied just as much in their path length as when compared to nodes in completely different countries.

Having in mind the fact that most nodes are located within Europe, it makes sense that route lengths, while naturally being different from node to node, don't vary as much as needed for this differentiation. Perhaps, with more nodes that are distributed all over the world, the path length in IP hops would have been a more significant decision criterion.

Luckily, the top level domain of each node's host name can be used to differentiate nodes into their estimated home countries. Figure 5.7 compares a set of different box plots, showing two at a time for one group of nodes. The red box on the left is the box plot for the plain Internet's route and the blue box on the right symbolizes the values measured through the overlay network. Lastly, the gray bar in the middle of each pair indicates the numbers of hosts in the particular group.

This short description and analysis of the plotted boxes in 5.7 briefly explains the information contained and possible reasons why the result may look different than expected. From left to right:

- **German** nodes are connected to the university's network and therefore to the web server using the X-WiN network administrated by the Deutsches Forschungsnetz association. Of course, this is also the group of nodes with the lowest physical distance to the web server, which is, combined with the good connectivity, the main reason why general performance is already very good and overlay routing can only add a delay to the packets because of the extra processing time needed.

  Additionally, most universities in Germany should be able to reach the web server without ever leaving the X-WiN/DFN or one of its regional allies such as the Leibnitz-Rechenzentrum in Munich, but sending packets to an Akamai server first may require leaving and reentering the DNF. Remembering the difficulties in peering contracts, switching between different autonomous systems makes the overlay network perform even worse in comparison.

- The **French** nodes surprisingly show a rather bad median RTT of around 80 ms,

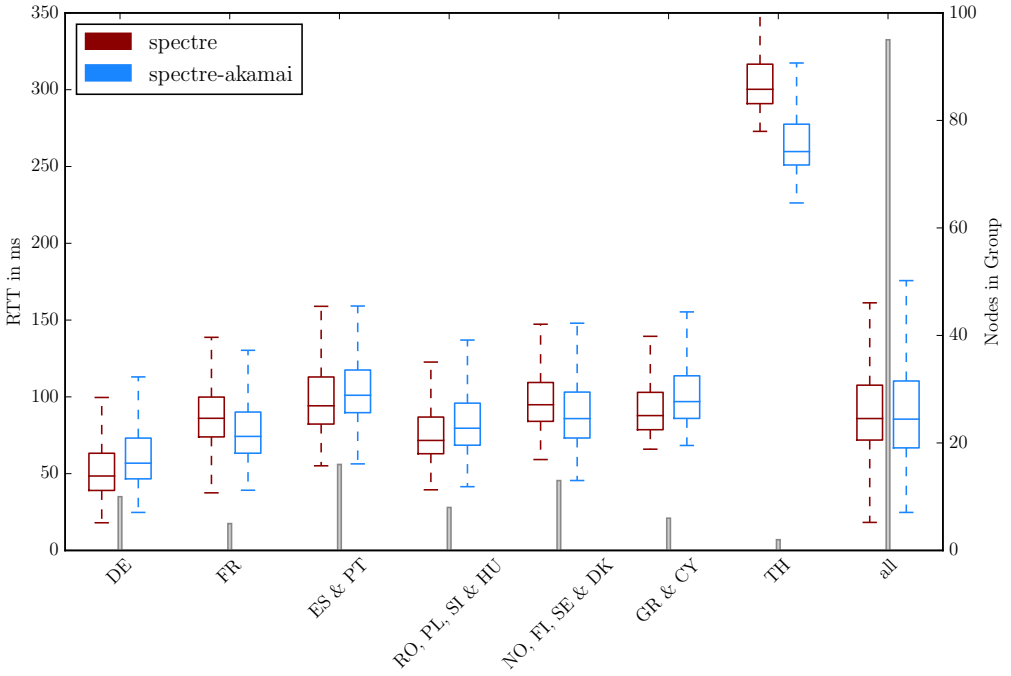


Figure 5.7: Box plot of selected node groups' round trip times

compared to below 49 ms for the German nodes. This, and the drastic improvement granted through overlay network routing, is a very unexpected result, because in general, connectivity to the German scientific network should already be very good.

In contrast to the next group, this result suggests rather selective network inefficiencies at the universities hosting the French PlanetLab nodes, which only feature a low number of nodes in total.

- Compared to the French nodes, connectivity **Spain** and **Portugal** is again slightly worse—just as it can be expected because of the higher geographic distance. Similar to the the nodes in Germany, overlay routing can only add a few milliseconds of delay to the packets, which is again surprising because even from France there is a benefit in round trip times when choosing the overlay route.

- The same holds true for the nodes in **Romania**, **Poland**, **Slovenia** and **Hungary**. Interestingly, median Internet latency is around 10 ms lower than from the French nodes. However, the distance to most of the eastern European countries is lower in reality than perceived in the minds of the people. For example, Ljubljana (Slovenia) is only 400 km away from Munich by car, Budapest (Hungary) around 670 km and Warsaw (Poland) around 1000 km. Madrid (Spain), on the other hand, is nearly 2000 km away from Munich and Lisbon (Portugal) even farther.

Of course, Internet routes don't follow the European highways and there are a many more factors influencing latency than just the geographic distance, which can only act as an indicator but nothing more. In the end, this results may just not be as surprising as it seems at first.

- Nodes from northern Europe in **Norway**, **Finland**, **Sweden** and **Denmark** behave as expected given their distance to the web server. The overlay network achieves to reduce latency by about 10 ms, which is a solid improvement for this distance.
- The nodes in **Greece** and **Cyprus** show a slightly better median plain Internet latency than the Nordic states. Also, overlay network routing can not improve latency and only adds its static overhead. Once again, these findings suggests a good connectivity of scientific networks between Germany and Greece.
- Fortunately, two nodes in **Thailand** were part of the PlanetLab Europe project and were widely available for measuring latency. Being the location with the highest distance to the web server, it is not surprising to see the high overall latency and next to the solid improvement made by overlay routing. After all, median latency through he overlay network is over 40 ms lower, which equals a relative advantage of around 15 %.
- Finally, all numbers of all nodes have been combined and plotted into one single box for each transport route. It turns out that overlay routing only features a very slight advantage over the plain Internet routing and in general increases the deviation of measurement values because of the problem with network jitter described above.

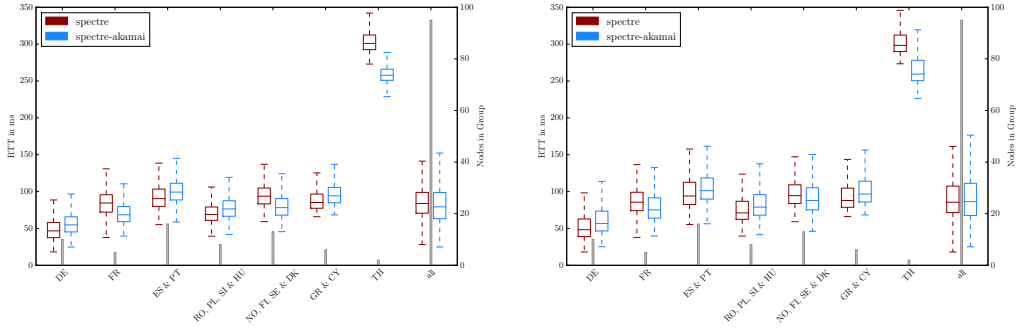
Analysis by Geographic Location and Inflicted Network Jitter

While the previous section used all data collected over the measurement period, it still is interesting to compare the different nodes groups with and without the network jitter. To do that, segments were selected just as in figure 5.6 but instead of regression lines, the box plots seen in figure 5.7 were plotted.

Figure 5.8 shows data from segments that did not experience any jitter on the left, and the ones containing significant outliers on the right. The box plots show a very consistent and also expected behavior:

In times of good network conditions, both boxes are about the sames size which means that using the overlay routing either improves the round trip times for nodes in that region or it can only add a small overhead, which consistently affects all 25^{th} percentile, the median and 75^{th} percentile.

In the other periods where a lot of jitter was observable, the quartile lines for both



(a) Only segments without jitter

(b) Only segments with jitter

Figure 5.8: Box plot of selected node groups' round trip times with and without network jitter

routes are drawn farther apart, so RTTs are more spread. Furthermore, in the rightmost group of the left graphic, the third quartile lines are exactly on par. In the jitter affected data however, there is a small disadvantage observable for overlay network routing in the third quartile line, although median RTT is the same.

Measuring worse RTTs over the overlay network than over plain Internet routing is actually the same phenomenon that was discussed in the section *5.3.2*, where one possible explanation was given. This comparison supports the theory presented there, because the small overhead only seems to exist in periods of bad network stability, but in the ideal case, it doesn't.

5.3.3 Conclusion

In practice, services such as Akamai's SureRoute would typically include more optimizations, for example on-the-fly compression and others, which would improve the actual performance for real world use-cases. Since these are not part of this thesis and would produced misleading results, these optimizations been turned off so that the round trip time differences on their own can be observed.

Concluding from the comparison of network routes in 5.7, it can be said that there in fact is a benefit that can be gained by simply[7] choosing a different route than the one advertised over BGP, depending on the location of test nodes and web server. For longer geographic distances, the routing benefit is clearly visible while for short distance connections the initial overhead can even produce slightly larger round trip times.

The benefit, while being clearly visible for certain locations, is a little less significant than originally assumed by the author. Keeping in mind that most of the nodes used

[7] *Simply* actually understates the technical difficulty of measuring Internet routes' RTTs and enabling accelerated routed through overlay networks on a larger scale, of course.

for measuring the latency were hosted by universities that are interconnected using exclusively scientific networks however, the results definitely show a solid improvement when performing HTTP GET requests against an overlay network routing accelerated web server.

5.4 File Download Measurements

Sending small amounts of data with the expectancy of low round trip times over persistent connection certainly has multiple applications, but not all data being transmitted is below 1 kB so it can fit into one packet just as in the previous experiment. For example, a typical use case for overlay network acceleration would be the delivery of a customer-tailored page for an on-line shop displaying individual items based on prior shopping behavior. The data could be generated in a central server cluster where access to the database is fast and then served to the client.

In this experiment, this scenario is modeled by requesting 100 kB of random data using regular **HTTP GET** requests, performing the full connection handshake and teardown.

5.4.1 Measurement Overview

The experiment took place over the period of 140 hours with the same nodes used in the first two experiments. These nodes would use `curl` to download 100 kB of data every 30 seconds over both plain Internet and overlay network routes and log the duration of the download to a text file. This process is continuously restarted every two hours, as in the previous experiments. In addition to the total download time, the measurement process also captured the time until the TCP connection was established. These times were measured by `curl` itself and can be output by using the `--write-out` option.

Over the time, 79 nodes were able to collect a total of 4993 segments containing download durations of 2 396 640 requests in total. As a side-note, this implies that 239.664 GB of payload data has been sent by the web server, half over plain Internet routing and half over the Akamai overlay network.

Overall, the success rate of delivered segments per time is about 15 % lower than in the second experiment, but still enough to conclude reasonable statements.

5.4.2 Results

The comparison of round trip times for small data units in *Accelerated Route Measurements* showed a mixed effects on round trip times where benefits where possible to

achieve for certain node groups. However, the differences were rather small and often times, overlay routing even inflicted a small overhead.

When measuring download times for larger objects as they would be typical on the web, even one single look at figure 5.9 reveals a much more significant benefit. Across the whole measurement period, download times of requests made over the Akamai overlay network are much faster than over the plain Internet routing.

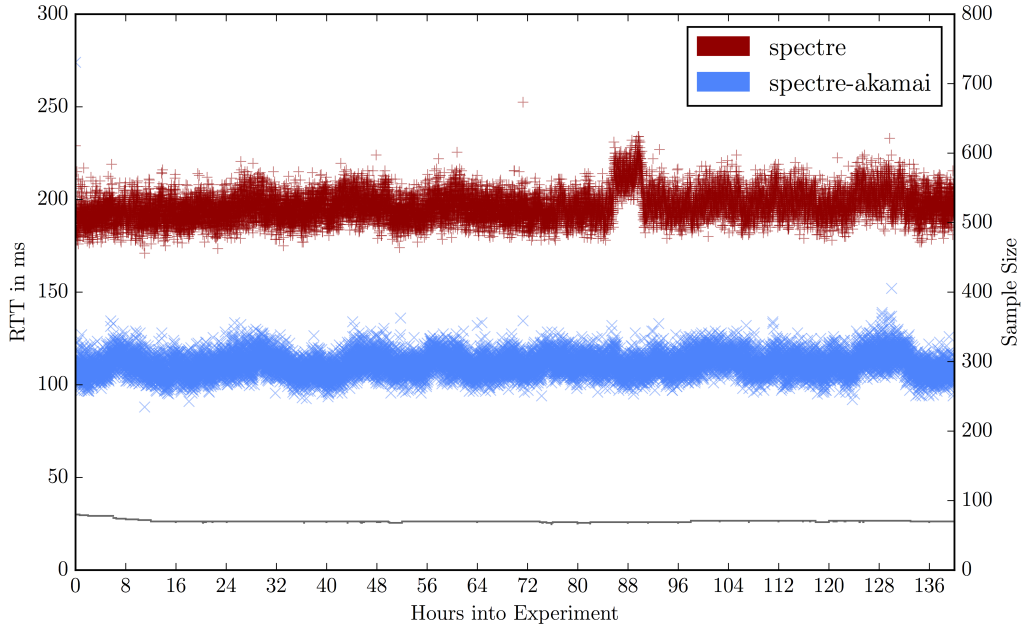


Figure 5.9: Scatter plot of 100 kB total download times median values

Noticeably, there are no severe network problems like jitter observable in the graph, except for a few sections where times are up to 10 % higher than usual for the direct Internet route. Next to the possibility that there were in fact no disturbances on the network, there is another difference to the previous experiment: Transferring 100 kB of data requires the use of around 69 packets, possibly more depending on the path's MTU and used header options. Therefore, performing even only one request already gives an average over many individual packets so that individually occurring jitter gets absorbed.

Since the overlay network's median values are drastically lower than plain Internet's, figure 5.10 shows the same measurement but corrected for the time needed to establish the TCP connection. That means that only the pure transfer duration is shown, but not the time it takes to perform the TCP handshake for each request.

It can be seen that when compensating the time needed for the TCP handshake, the resulting transfer times are much more closely together. Still, the overlay networks achieves the delivery faster throughout the whole measurement.

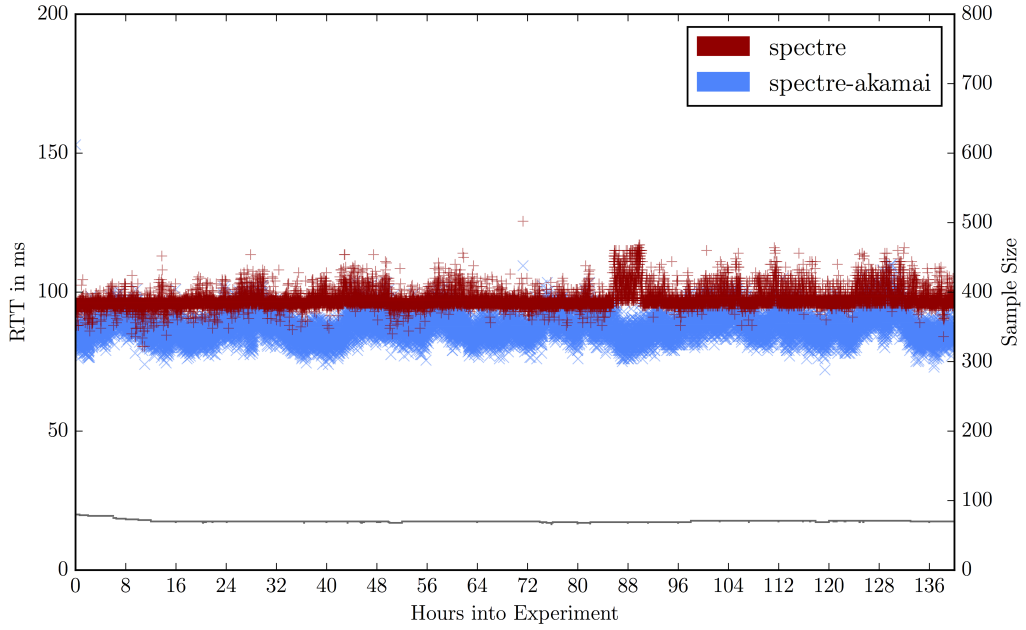


Figure 5.10: Scatter plot of 100 kB transfer only times

In order to compare the different node groups introduced in 5.3.2, the same box plot is shown in figure 5.11. The graph essentially shows a very consistent improvement of transfer times across all node groups, even the ones that were not accelerated by overlay network routing for single packets.

It is surprising that even nodes in Germany, which should not really profit from overlay routing, do so according to the figure. While this would be reasonable for most node groups that have the potential to have their routes optimized, this advantage for the German nodes suggests, that the experiment did in fact not only measure routing benefits but other influences, too.

The earlier chapters of this thesis already described the optimizations used by Akamai and mentioned that the *TCP optimizations* in general have been turned off. Unfortunately, it can't be confirmed that default values and default procedures are still in place that accelerate the use of TCP connections between the Akamai servers. That means, even without the explicit use of TCP optimizations, it is possible that the congestion window's size and similar parameters are recorded and re-used between the periodic requests.

Avoiding the typical TCP slow-start phase would result in a measurable benefit for overlay network routes when compared to completely new connections over the plain Internet. Since these factors cannot be completely excluded, the data of this experiment is not suitable to investigate routing differences any further.

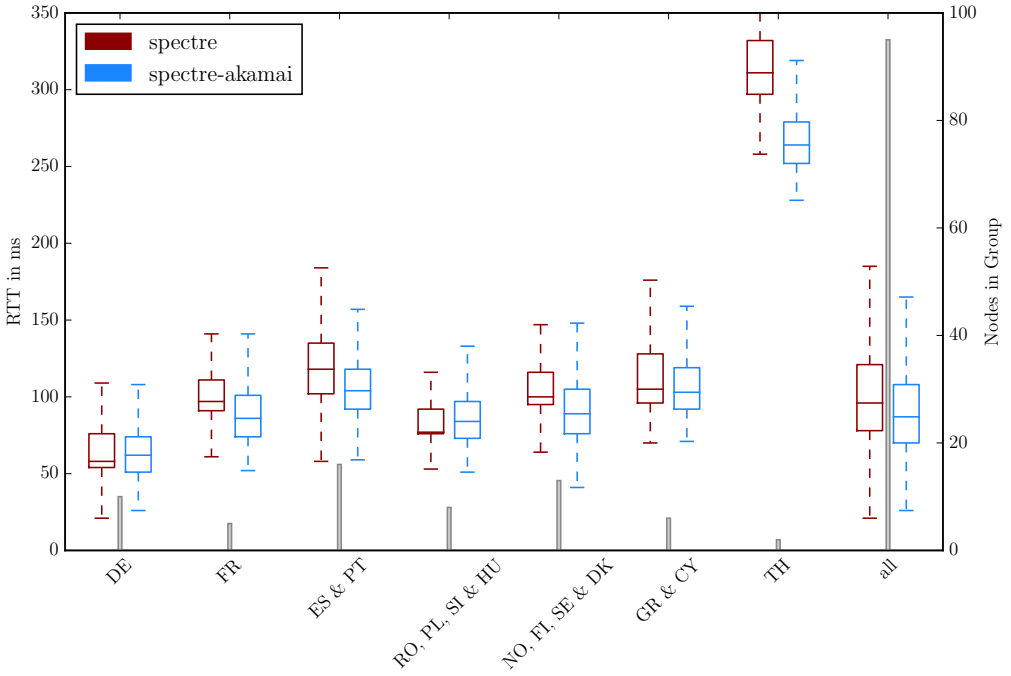


Figure 5.11: 100 kB transfer only times by node group

5.4.3 Conclusion

As mentioned above, it cannot be said with absolute certainty, that the benefits in download times of the 100 kB over the overlay network are solely founded by possible routing advantages.

Separating the TCP connection times from the total download duration gives a hint towards the explanation of the large difference in download times. It can be seen that the distance between client and first server plays an important role because of the initial RTT required to perform the TCP handshake. Deploying a large overlay network with nodes at strategic locations can surely help accelerating this aspect.

Even without more aggressive TCP optimization settings, there is a clear advantage in the overlay network delivery. This shows that, for transferring larger amounts of data as in this experiment, there are effective techniques in addition to using an optimized routed, especially in the area of tweaking TCP parameters for more effective delivery.

In practice, on-the-fly compression and caching with proactive requests of referenced resources could bring even more acceleration than seen in this experiment.

5.5 Case Studies

To finish the exploration of overlay routing, three case studies have been made. The aim is to gain more insight into why certain locations benefit more from overlay routing than others by comparing their routes to the web server.

The `traceroute` tool was used to gather all IP hops between the client and the web server over the plain Internet and also following the route packets take when accelerated by Akamai SureRoute. Obtaining these traces is usually not possible for neither clients nor Akamai customers, but was thankfully permitted for this scientific exploration.

The process of obtaining traces over the Akamai network starts with initiating a regular HTTP request to the web server over the overlay network. It then involves looking up that specific request in the HTTP logs of the first Edge Server that was handed out to the client via DNS. The log file contains information revealing the next hop that was chosen, which can be either the origin server itself or another intermediate Akamai server. This manual lookup can be repeated from Edge Server to Edge Server until all the *anchor points* of the overlay network's route are found.

After that, an Akamai tool also available to paying customers can be used to perform a `traceroute` between each pair of Edge Servers on the route that was identified beforehand. That way, the actual IP level trace can be assembled and compared with the one measured by the client itself.

The tables showing the individual hops also include AS number, network provider and geographic location, where possible. For performing AS lookup, the `whois.cymru.com` whois-server was used[8] and geographic locations where resolved using the *MaxMind GeoIP2* demo website[9]. Both sources were augmented and checked for consistency with Akamai internal data sources not available to the general public, where needed.

It is still possible that hops are out of alignment with the geographic location or the autonomous system displayed and actually belong to the corresponding entry in the row above or below. However, the routes themselves should be respectably represented.

5.5.1 Germany

For the first route analysis, a node in Germany is chosen. This group of nodes was expected to show low round trip times over plain Internet routing without any gain by overlay routing and *Accelerated Route Measurements* already confirmed that assumption. The web server is a logical part of the chair I8's network at TU München, so all packets are bound to traverse the universities network. For this network, upstream is provided

[8] `http://www.team-cymru.org/IP-ASN-mapping.html`
[9] `https://www.maxmind.com/de/geoip-demo`

by the Leibnitz Rechenzentrum in Munich which in turn is connected to the X-WiN/DFN scientific network that spans over nearly all of Germany[10].

The node chosen as a client for the traceroute measurements was the PlanetLab node `mars.planetlab.haw-hamburg.de` at the *Hamburg University of Applied Sciences* (HAW Hamburg).

Plain Internet Route

Table 5.5 shows the route as measured from the client to the web server using standard tools and default Internet routing.

#	IP	AS	RTT	Estimated Location
1	`141.22.213.33`	680	0.535	HAW Hamburg
2	`141.22.4.148`	680	0.360	HAW Hamburg
3	`188.1.231.165`	680	0.865	X-WiN/DFN, Hamburg
4	`188.1.144.1`	680	0.445	X-WiN/DFN, Hamburg
5	`188.1.146.145`	680	5.053	X-WiN/DFN, Hanover
6	`188.1.144.141`	680	14.172	X-WiN/DFN, Frankfurt
7	`188.1.37.90`	680	20.933	X-WiN/DFN, Munich
8	`129.187.0.150`	12816	20.893	LRZ Munich
9	`131.159.252.1`	12816	20.617	TU Munich
10	`131.159.252.150`	12816	20.413	TU Munich

Table 5.5: Plain Internet route from Hamburg to Munich

Given the short distance, the route has relatively many hops, as the comparison with the other cases will show. However, the first and last two hops are within the nodes' own universities' networks and don't add significant latency to the previous hop, as the table shows.

Apart from the high quota of short-distance hops like the first two and the last four, the route offers no interesting findings—the route goes as expected from north to south across Germany.

Overlay Network Route

Retrieving the complete overlay network route is a little more complicated, as explained above. Examining the route from client to the first Edge Server is of course no problem, the second half from Edge Server to `spectre.net.in.tum.de` on the other hand had to be done via Akamai internal tools.

[10] `https://www.dfn.de/fileadmin/3Beratung/Betriebstagungen/bt62/Plenum-Neues_zum_X-WiN.pdf`

Table 5.6 shows the compound route from the same node in Hamburg to the web server in Munich using the overlay network, the highlighted row marking an Akamai Edge server. The round trip times are absolute values as measured from the PlanetLab client. For route segments measured beginning at an Akamai Server, they were added to the previously measured RTT to produce a consistent series, but without adding processing and computation time that would be usually needed for the Akamai Edge Servers to choose the following intermediate target. It can be seen that hops later on the path may feature lower RTTs than previous hops, because only few RTTs measurements were made for this comparison and the results are therefore of limited significance.

As requested by Akamai, the IP addresses of Akamai deployments are not disclosed.

#	IP	AS	RTT	Estimated Location
1	`141.22.213.33`	680	0.412	HAW Hamburg
2	`141.22.4.148`	680	0.366	HAW Hamburg
3	`188.1.231.165`	680	0.858	X-WiN/DFN, Hamburg
4	`188.1.144.1`	680	0.448	X-WiN/DFN, Hamburg
5	`188.1.146.145`	680	5.185	X-WiN/DFN, Hanover
6	*(filtered)*	680	4.156	Akamai Edge Server, Hanover
7	*(filtered)*	680	5.764	Akamai Deployment, Hanover
8	`188.1.144.141`	680	14.761	X-WiN/DFN, Frankfurt
9	`188.1.37.90`	680	22.294	X-WiN/DFN, Munich
10	`129.187.0.150`	12816	22.262	LRZ Munich
11	`131.159.252.1`	12816	28.746	TU Munich
12	`131.159.252.150`	12816	21.757	TU Munich

Table 5.6: Overlay network route from Hamburg to Munich

The table immediately reveals the enormous resemblance of the overlay network's route to the plain Internet route. Only hop number 6 and 7 are used exclusively in the overlay network route and originate in the fact that the client has to initially connect to one Akamai Edge Server which decides over the following routing procedure. Because of the already optimal route to the web server in Munich, the Edge Server decides not to route the packets over another Edge Server closer to the target, but instead just uses the plain Internet route again.

It can also be seen that the exact same autonomous systems are used to deliver the packets. This circumstance would usually reduce the overhead from using overlay routing because fewer AS-transitions have to be made, but in this case there really is no improvement to be made over the existing route.

Conclusion

In addition to the tables, the routes are graphically compared in figure 5.13.

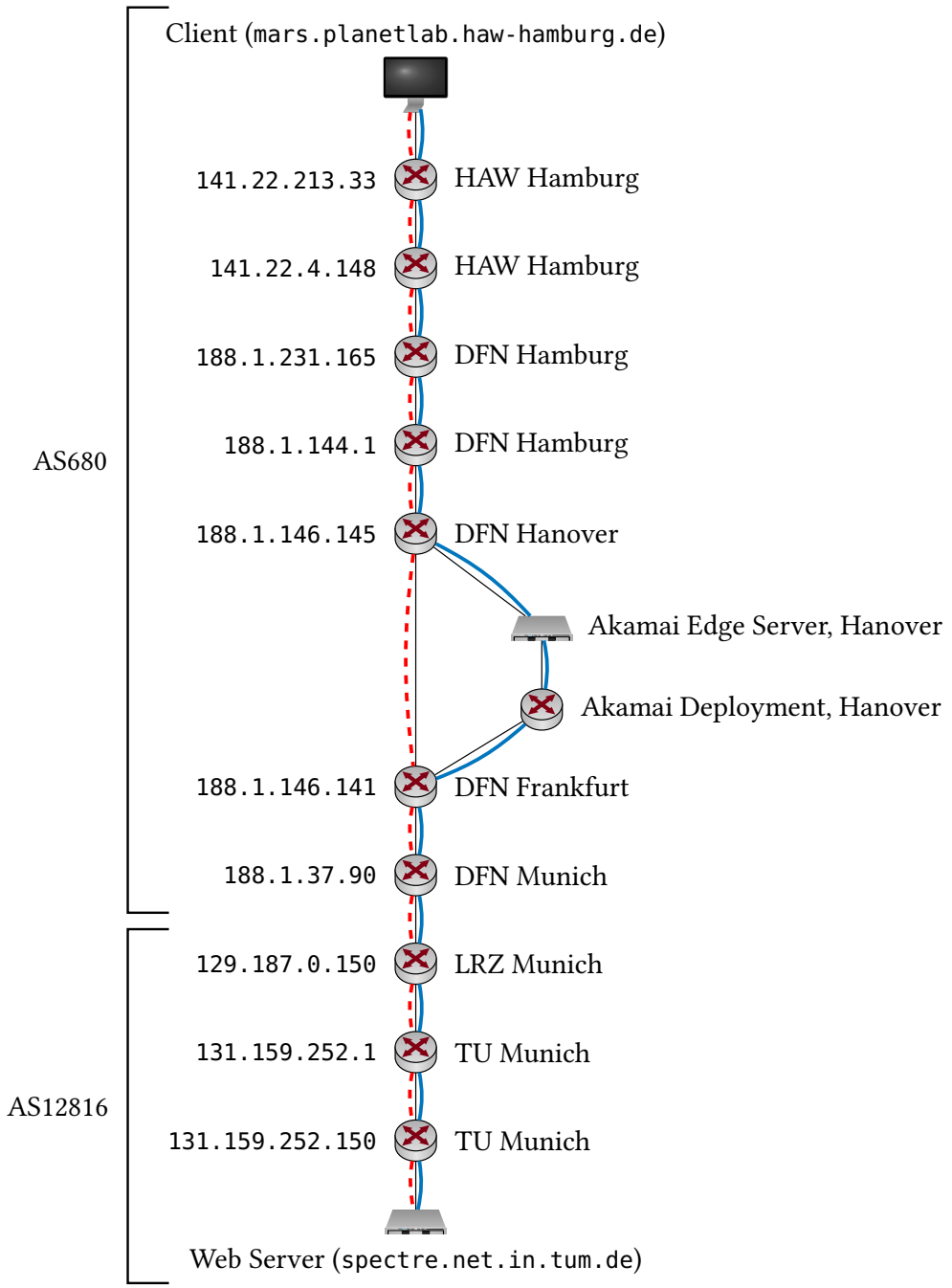


Figure 5.12: Graphical comparison of regular Internet and overlay network route from Hamburg to Munich

Because of the good connection between the client and the web server through the X-WiN/DFN, the first and the last 5 hops are exactly the same. By including two extra hops, one of which of course is not a router but an Akamai server, this route is a good example of why overlay routing adds a small overhead for clients in this region. Summing up

the RTTs from client to Edge Server and Edge Server to web server even leads to nearly the same value for this overhead that has been measured in the first experiment.

Studying this route, while not resulting in any noteworthy insights, provided a plausible explanation of the values measured earlier. Also, it illustrates the principles of route analysis on a simple example.

5.5.2 France

A possibly more interesting result was received from the nodes in France. Figure 5.7 showed that the overlay network achieved a solid improvement of round trip times compared to regular Internet routing, which had proven to be unexpectedly bad, given the relatively low distance between the nodes and the web server.

While there actually were nodes that did not benefit from overlay routing, this case study focuses on one that is more typical for the group's box plot. The node used for traceroute measurements is `plewifi.ipv6.lip6.fr` located at the *Pierre and Marie Curie University* (UPMC) in Paris.

Plain Internet Route

As with the node in Germany, a simple `traceroute` measurement was enough to find out all the IP hops between the node and the web server on the regular Internet route. Table 5.7 contains the results of this traceroute.

#	IP	AS	RTT	Estimated Location
1	`132.227.62.65`	1307	0.477	UPMC, Paris FR
2	`10.1.1.1`	1307	1.938	Internal Router
3	`134.157.167.125`	1307	2.676	UPMC, Paris FR
4	`134.157.254.124`	1307	3.396	UPMC, Paris FR
5	`195.221.127.181`	2200	2.530	RENATER, Paris FR
6	`193.51.181.102`	2200	2.978	RENATER, Paris FR
7	`193.51.177.117`	2200	2.220	RENATER, Paris FR
8	`193.51.177.24`	2200	6.810	RENATER, Paris FR
9	`62.40.124.69`	20965	2.923	GÉANT, Paris FR
10	`62.40.98.76`	20965	8.369	GÉANT, London GB
11	`62.40.98.81`	20965	15.987	GÉANT, Amsterdam NL
12	`62.40.112.146`	20965	32.429	GÉANT/DFN, Berlin DE
13	`188.1.144.186`	680	42.543	X-WiN/DFN, Erlangen DE
14	`188.1.241.194`	680	45.874	X-WiN/DFN, Garching (Munich)
15	`131.159.252.1`	12816	45.938	TU Munich
16	`131.159.252.150`	12816	45.183	TU Munich

Table 5.7: Plain Internet route from Paris to Munich

The route starts at the UPMC where on hop 2, a probably misconfigured router sends ICMP messages containing a private IP address. After the first 4 hops, the national research network *RENATER* is entered which adds another 4 hops inside Paris before the European research network *GÉANT* is entered. Until that point, not at lot of time was added to the RTTs measured, although the values vary a lot from hop to hop.

The interesting part of the route is the way traffic is routed from Paris from there on: While it can't be confirmed with absolute certainty, packets seem to be routed over London, Amsterdam and Berlin before entering the German scientific network X-WiN/DNF. The increasing round trip times as well as the *GÉANT* topology map[11] support this information and make this measurement more reasonable.

Using location data from the *MaxMind* database only, the questionable IP hops can only be located as shown in table 5.8. It can be seen that the location data for all hops in the *GÉANT* and *W-WiN/DFN* is imprecise and the entries in 5.7 therefore have to be handled with care.

#	IP	AS	RTT	Estimated Location
⋮				
9	`62.40.124.69`	20965	2.923	GÉANT, GB
10	`62.40.98.76`	20965	8.369	GÉANT, GB
11	`62.40.98.81`	20965	15.987	GÉANT, GB
12	`62.40.112.146`	20965	32.429	GÉANT/DFN, DE
13	`188.1.144.186`	680	42.543	X-WiN/DFN, DE
14	`188.1.241.194`	680	45.874	X-WiN/DFN, DE
⋮				

Table 5.8: Excerpt of plain Internet route from Paris to Munich using only MaxMind location data

After entering the DFN, the route to Munich is rather simple and comprehensible, including its round trip times and intermediate hops, given the accuracy of its location information.

Overlay Network Route

In order to compare the overlay network's route, the same process as described above was used. Table 5.9 shows this route, again highlighting the Akamai Edge Servers that did actively choose the next intermediate hop.

Round trip times are again in absolute values but not containing possible processing delay introduced by the Akamai servers.

[11] `http://www.geant.net/Resources/Media_Library/Pages/Maps.aspx`

#	IP	AS	RTT	Estimated Location
1	`132.227.62.65`	1307	0.220	UPMC, Paris FR
2	`10.1.1.1`	1307	0.941	Internal Router FR
3	`134.157.167.125`	1307	2.034	UPMC, Paris FR
4	`134.157.254.124`	1307	1.978	UPMC, Paris FR
5	`195.221.127.181`	2200	1.508	RENATER, Paris FR
6	`193.51.181.102`	2200	2.214	RENATER, Paris FR
7	`193.51.177.114`	2200	6.441	RENATER, Paris FR
8	*(filtered)*	2200	2.180	Akamai Deployment, Paris FR
9	*(filtered)*	2200	2.460	Akamai Edge Server, Paris FR
10	`193.51.224.33`	2200	3.846	RENATER, Paris FR
11	`195.10.62.25`	1273	3.121	CW Cable and Wireless, Paris FR
12	`195.2.22.166`	1273	3.121	CW Cable and Wireless, Paris FR
13	`64.125.27.78`	6461	11.789	AboveNet, Frankfurt DE
14	`94.31.37.122`	6461	11.850	AboveNet, Frankfurt DE
15	*(filtered)*	20940	11.660	Akamai Edge Server, Frankfurt DE
16	*(filtered)*	20940	11.826	Akamai Deployment, Frankfurt DE
17	`94.31.37.121`	6461	11.873	AboveNet, Frankfurt DE
18	`80.81.192.222`	6695	23.013	DE-CIX/DFN, Frankfurt DE
19	`188.1.37.90`	680	20.304	X-WiN/DFN, Munich DE
20	`129.187.0.150`	12816	20.309	LRZ Munich
21	`131.159.252.1`	12816	20.075	TU Munich
22	`131.159.252.150`	12816	19.969	TU Munich

Table 5.9: Overlay network route from Paris to Munich

The first 6 hops are exactly the same as on the public Internet route. After that, an Akamai deployment hosted within the academic network *RENATER* is contacted instead of routing traffic to the *GÉANT* network, which, in fact, is not used at all on this route.

The Akamai Edge Server at 9 chooses to route the packets to another Edge Server (hop 15) in Frankfurt and for that purpose uses the *Cable & Wireless Worldwide*[12] infrastructure. In Frankfurt, the Akamai deployment is hosted by *AboveNet*[13], which is why the hops before and after 15 belong to AS 6461. AS 20 940, on the other side, is an AS owned by *Akamai International* which is used to host Akamai Servers and work with the BGP announcement stream at various strategic locations on the Internet.

After the hops in Frankfurt, the route to Munich is again straightforward, even though hop 18 can't be identified and explained without a doubt.

[12] Acquired by Vodafone in 2012
[13] Acquired by the Zayo Group in 2012

Conclusion

In addition to the tables, the routes are graphically compared in figure 5.13.

This case study shows an example client where overlay routing does not only yield a certain round trip time benefit, but where the routes taken are drastically different.

Routing traffic from France over London can make sense when seen from the architectural side of the *GÉANT* network, because the focus may be on throughput to enable the exchange on large amounts of data. Granted, latency is always an important aspect when designing networks, but the point is that there may exist meaningful reasons to route traffic the way it was done in table 5.7.

For this particular connection, a route that goes from Paris directly to Frankfurt, as seen in the overlay route, is clearly advantageous—even though it contains 6 more hops, two of which are actually not routers but servers needing additional computation time.

5.5.3 Thailand

The third case study involves the PlanetLab node `ple1.ait.ac.th` located at the *Asian Institute of Technology* (AIT) in the north of Bangkok, Thailand.

The two nodes at AIT showed the most significant improvement in round trip times for packets routed through the overlay network, which is not very surprising given the functioning of overlay routing. The initial presumption was that long routes can benefit more from overlay routing, and the RTT measurements seemed to confirm this point.

Plain Internet Route

The route used with regular Internet routing was found by running a `traceroute` measurement from the node in Bangkok to the web server in Munich. Table 5.10 shows this route.

The first two hops happen within the university's own network and therefore show very good round trip times. The path continues into first the *UniNet* and then *ThaiREN* network, which are both nationally deployed Thai research networks. This behavior is the same as in the other two case studies and of course an unsurprising choice in network architecture.

An interesting part starts at hop 6, where packets are handed over to *TEIN3*, the *Trans-Eurasia Information Network*. Even though ip-to-location lookups for hops 6 to 8 are indicating those three nodes would be located in Beijing, China, it is more likely that the nodes whose hostnames are `sg-ge-03-v4.bb.tein3.net`, `mb-so-01-v4.bb.tein3.net`

#	IP	AS	RTT	Estimated Location
1	203.159.63.254	4767	0.811	AIT, Bangkok TH
2	203.159.63.77	4767	0.631	AIT, Bangkok TH
3	202.28.214.45	4621	4.275	UniNet, Bangkok TH
4	202.28.218.6	4621	3.916	UniNet, Bangkok TH
5	202.29.12.13	24475	1.263	ThaiREN, Bangkok TH
6	202.179.249.65	24490	31.859	TEIN3, Singapore[a]
7	202.179.249.54	24490	90.846	TEIN3, Mumbai IN[a]
8	202.179.249.118	24490	201.997	TEIN3, Central Europe[ab]
9	62.40.98.67	20965	221.646	GÉANT, Zürich CH
10	62.40.98.109	20965	230.577	GÉANT, Frankfurt DE
11	62.40.124.218	20965	230.768	GÉANT/DFN, Frankfurt DE
12	188.1.37.90	680	238.476	X-WiN/DFN, Munich DE
13	129.187.0.150	12816	238.476	LRZ Munich
14	131.159.252.1	12816	250.019	TU Munich
15	131.159.252.150	12816	238.254	TU Munich

Table 5.10: Plain Internet route from Thailand to Munich

[a] GeoIP databases suggest Beijing, China, but hostnames and submarine cables suggest otherwise
[b] Hostname suggests Madrid, Spain, but stops of common submarine cables hint at Marseille, France

and `eu-mad-pr-v4.bb.tein3.net` are actually located in Singapore (SG), Mumbai (MB) and Madrid (MAD), although this last city could also easily be Marseille.

So even though the geoip lookups are not consistent with the very little information contained in the hostnames (which have to be handled with care, too), looking at common submarine cables[14] reveals multiple cables that have stops in Singapore, Mumbai and finally Marseille as the only ending point of presence that is reasonable in this context:

- SEA-ME-WE 4 (South-East Asia – Middle East – Western Europe)
- SEA-ME-WE 5
- AAE-1 (Asia – Africa – Europe)

After the route has reached the European main land, the traffic is handed over to *GÉANT* and takes a comprehensible route to Munich.

Overall, this route contains only 15 hops, one less than the plain Internet route from Paris to Munich, even though the greater distance is clearly visible in the latency which is many times as high as in the earlier case study. Still, it's a good example why path length in IP hops is no suitable metric for clustering nodes into groups for analysis like the one in section 5.3.

[14] `http://www.submarinecablemap.com`

Overlay Network Route

The plain Internet route from Bangkok to Munich did not reveal any great indirections as seen on the route from Paris to Munich. To find out why the overlay network accelerated route features lower round trip times nevertheless, table 5.11 shows the combined `traceroute` results over two Akamai Edge Servers.

#	IP	AS	RTT	Estimated Location
1	`203.159.63.254`	4767	0.733	AIT, Bangkok TH
2	`203.159.63.77`	4767	0.678	AIT, Bangkok TH
3	`202.28.214.45`	4621	1.824	UniNet, Bangkok TH
4	`202.28.218.22`	4621	3.038	UniNet, Bangkok TH
5	`122.155.253.161`	N/A	2.057	Thailand
6	`122.155.253.230`	N/A	3.888	Thailand
7	`203.144.193.81`	7470	3.706	TrueInternet, Bangkok TH
8	*(filtered)*	7470	3.669	Akamai Deployment, Bangkok TH
9	*(filtered)*	7470	3.463	Akamai Edge Server, Bangkok TH
10	*(filtered)*	7470	3.977	Akamai Deployment, Bangkok TH
11	`61.90.191.142`	7470	3.971	TrueInternet, Bangkok TH
12	`203.144.144.10`	7470	4.108	TrueInternet, Bangkok TH
13	`103.3.177.190`	7470	4.745	TrueInternet, Bangkok TH
14	`122.144.25.193`	38082	7.631	True Internat. Gateway, TH
15	`113.21.245.110`	38082	33.523	True Internat. Gateway, SG
16	`180.87.96.29`	6453	32.867	TATA, Singapore
17	`180.87.96.22`	6453	210.579	TATA, Mumbai IN
18	`180.87.15.146`	6453	209.589	TATA, Mumbai IN
19	`80.231.217.17`	6453	209.970	TATA, Marseille FR
20	`80.231.217.2`	6453	211.278	TATA, Marseille FR
21	`80.231.200.78`	6453	211.940	TATA, Frankfurt DE
22	`195.219.87.18`	6453	212.921	TATA, Frankfurt DE
23	*(filtered)*	6453	210.389	Akamai Deployment, Frankfurt DE
24	*(filtered)*	6453	208.583	Akamai Edge Server, Frankfurt DE
25	*(filtered)*	6453	210.421	Akamai Deployment, Frankfurt DE
26	`195.219.50.193`	6453	208.967	TATA, Frankfurt DE
27	`195.219.50.50`	6453	209.118	TATA, Frankfurt DE
28	N/A	N/A	N/A	N/A[a]
29	`212.162.4.6`	3356	210.565	LEVEL3, Frankfurt DE
29	`188.1.37.90`	680	217.613	X-WiN/DFN, Munich DE
30	`129.187.0.150`	12816	218.077	LRZ Munich
31	`131.159.252.1`	12816	228.992	TU Munich
32	`131.159.252.150`	12816	217.476	TU Munich

Table 5.11: Overlay network route from Thailand to Munich

[a]No reply was recieved from this router

It is apparent even at first sight, that the route features an enormous number of IP hops—more than the default limit of 30 in most `traceroute` implementations.

The route starts as expected until hop 4 and even though hops 5 and 6 were couldn't be identified, it is clear from that fact that the Akamai Edge server in 9 is still in Bangkok, that 5 and 6 surely are within Bangkok, too. For this route however, the Akamai deployment happens to not be within the national scientific network like in Germany and France, but hosted at a larger national ISP, *TrueInternet*.

The Edge Server decides to route the traffic to Frankfurt, which is a not very surprising fact, and packets travel through the international gateway of *TrueInternet* in order to reach the network of *Tata Communications*. The hops from number 16 to 19 are very clearly following the same route as the plain Internet, probably using one of the submarine cables listed above. After landing in France, the data is routed to the next Akamai Edge Server in Frankfurt (24), completely without leaving the autonomous system.

The packets only have to change network after the next routing decision in Frankfurt, where they are handed over to *Level 3*, which is one of the upstream providers for the X-WiN/DFN[15].

Conclusion

In addition to the tables, the routes are graphically compared in figure 5.14 where the Akamai deployments are compressed into one line each for the sake of brevity.

This case study shows that gaining a round trip time advantage does not necessarily require finding a complete new physical way to the target. It can only be speculated why the plain Internet route through the academic networks is significantly slower in RTTs, perhaps there was more load on the routers and inter-network transit points. But again, this cannot be said with certainty and is open for discussion.

5.5.4 Summary

The case studies compared routes in completely three different situations:

- **German nodes** already had good round trip times over regular routing and only a few milliseconds of overhead were added by the overlay routing.

  This results conforms with the expectancy of direct and already efficient routes not being able to gain any benefit through route optimization.

[15] https://www.dfn.de/fileadmin/3Beratung/Betriebstagungen/bt62/Plenum-Neues_zum_X-WiN.pdf

- **French nodes** showed surprisingly bad RTTs for the plain Internet route, even though the nodes are geographically close to Germany and peering between the scientific networks is expected to be good.

  However, it turned out that traffic probably gets routed over London, Amsterdam and Berlin in order to reach Munich. It comes to no surprise that this route can be optimized and the overlay network showed the potential of a route that goes from Paris to Frankfurt and then to Munich, even though the scientific network had to be left in between.

- **Thai nodes** performed worst over regular Internet routing and gained a lot from overlay network routing.

  The case study showed that both paths generally take the same route, but the difference in network operator can of course result in differences in round trip times and throughput. Acceleration through overlay routing therefore comes from a different reason than in the French case study.

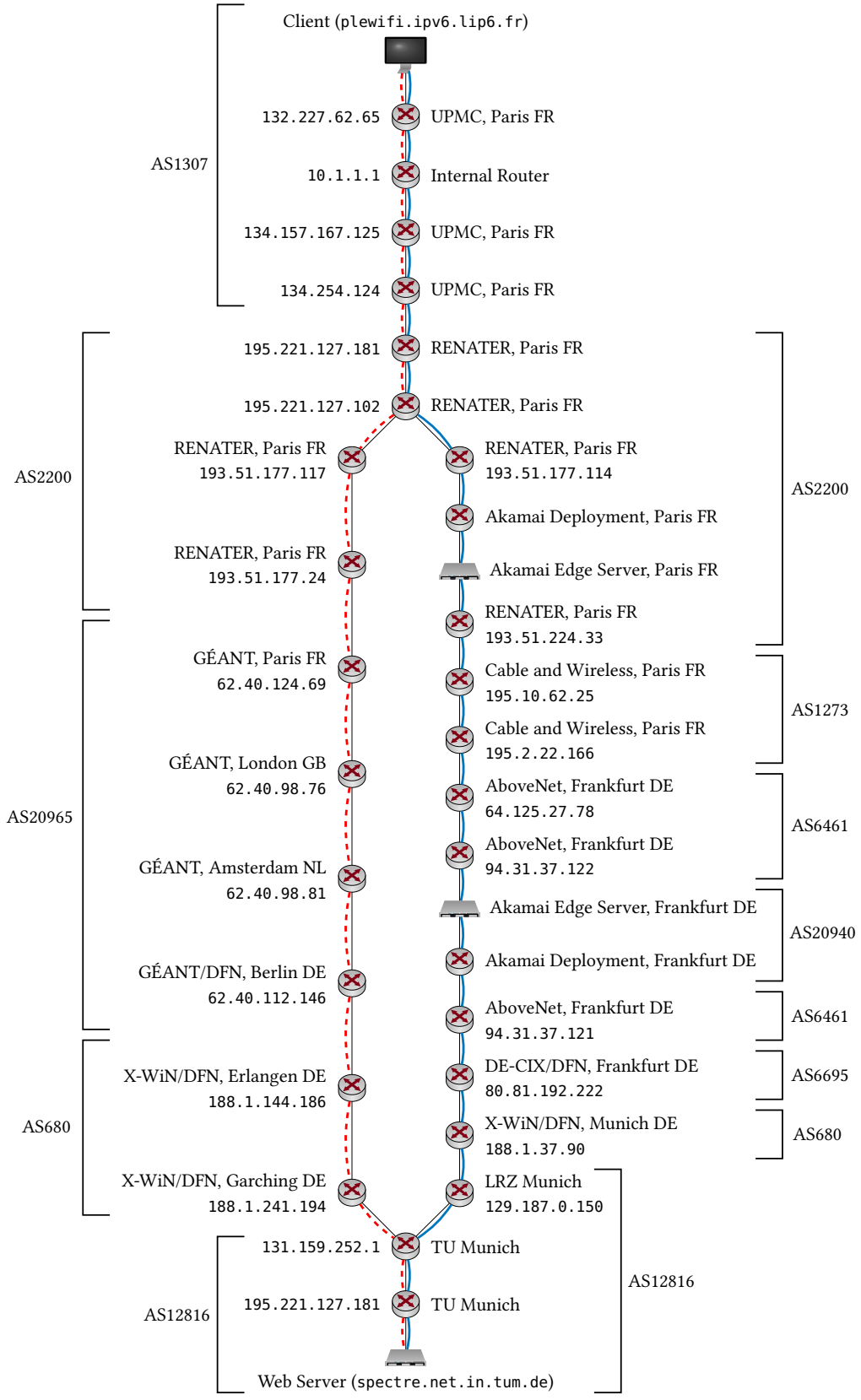


Figure 5.13: Graphical comparison of regular Internet and overlay network route from Paris to Munich

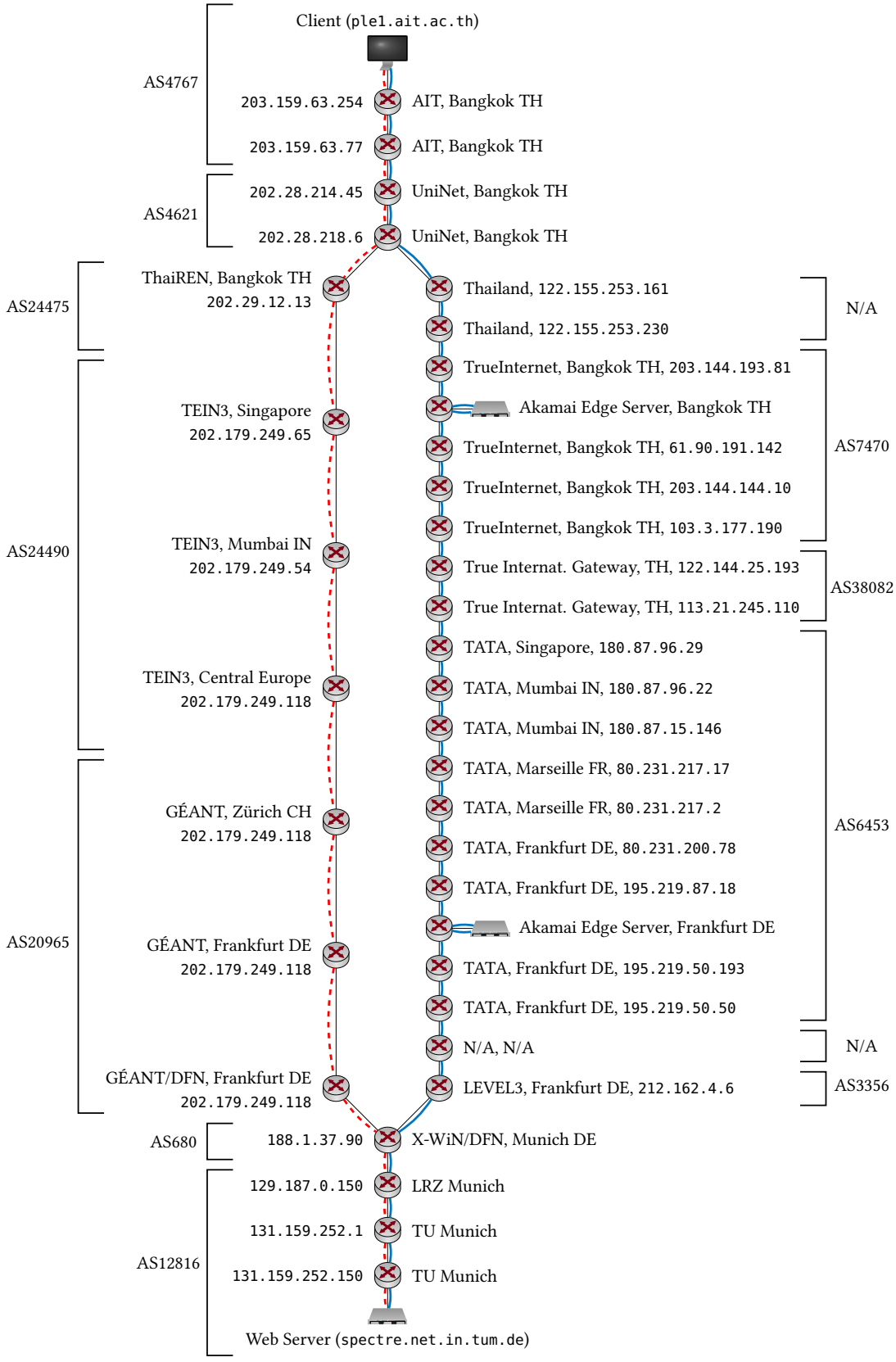


Figure 5.14: Graphical comparison of regular Internet and overlay network route from Bangkok to Munich

Chapter 6

Conclusion

Summarizing the results that were discussed in the previous section, this chapter will give answers to the research question posed in the beginning. Also, interesting prospects for future work that derives from the content of this thesis are given.

6.1 Future Work

Even though different experiments and case studies have been made, the research area of practical overlay networks and the effects of optimizing transport routes offers many more interesting topics that expand well beyond the area covered by this thesis. Some of these are listed below and hopefully inspire future research on this topic.

6.1.1 Overlay Networks

How can finding optimized routes be realized?

Designing and implementing an overlay network that has practical benefits over classic BGP routing is a complex task.

What metrics are to be collected, how to probe only the most promising ways instead of every connection through the Internet and how are results ranked in order to find the best route?

What effect does overlay network routing have on a providers upstream policies?

While network providers exchange traffic for free to their peering partners, upstream connections are paid for and one provider will never accept traffic targeted to one of

his peerings through its upstream.

Is is possible for overlay networks to leverage a potentially better route through a providers upstream connection, even though the client is not a customer of the operator itself, but instead customer of one of its peering partners' networks?

6.1.2 Overlay Network Measurements

Using a larger set of test nodes

During this work, it was possible to use test nodes that were distributed across Europe and a few at more distance locations. Usually, PlanetLab allows accessing nodes on a world-wide base, but at the time the experiments where made, there seemed to exist a problem with using PlanetLab Europe accounts to log onto servers belonging to PlanetLab Central.

When using a larger group of nodes, for example when using nodes that are distributed all over the world, the resulting comparison would show clearer differentiation when grouped by distance, country or continent. Also, more case studies could be made to explore the situation of different long-ranged connections.

Simulate link failure and measure reaction times

Companies like Akamai also advertise the enhanced availability of web services when the hosting data center or an intermediate connection experiences link failure, given a possible alternative route of course.

It would be an interesting experiment to simulate such a link failure and continuously measure the availability of the web server via both the regular Internet route and the overlay network route. How long does it take to recover and how much is the round trip time affected by this?

6.2 Answering the Research Questions

Coming back to the research questions from the beginning of this work, it can be said that all three questions have been answered through the experiments and measurements, which shall now be summarized.

How does optimized routing within an overlay network affect the latency of small sized data units?

This question was asked with uses cases in mind that required the rapid exchange of small sized data units, for example in collaborative live editing scenarios, but also for quick interactive web transactions over persistent connections.

As described in section 5.3, there has been a great number of requests made over a persistent TCP connection for both the regular Internet route and the Akamai overlay network. Since all other optimizations have been disabled, it can be said that, depending on the node location, there is a noticeable advantage for round trip times of routes using the overlay network to avoid congestion points on the Internet.

It turned out that the result heavily depends on the location of the test nodes chosen. For example, nodes within the same country as the web server did not benefit from overlay routing and instead experienced a small overhead of up to 5 ms originating in the extra hops required by connecting to an overlay network's server first. On the other hand, test nodes located far away received a round trip time benefit of up to 15 %, which can be quite notable improvement and represents a noteworthy result for just choosing another route and not applying any other optimizations.

These results were obtained using nodes primarily located on scientific networks that are expected to feature a good connection to other scientific networks and furthermore already use good routing schemta and links loaded well under maximum capacity. For regular clients on home Internet connections, there may be even greater benefits because it would only seldom be the case that content provider and client are connected to the same network. On top of that, routing choices or commercial network providers are affected stronger by economical decisions and policies, which can result in suboptimal indirections and inefficiencies.

How does optimized routing within an overlay network affect the download time of medium sized objects over HTTP?

This next questions aims towards the same optimization as in the first question, but for a slightly different use case. For users surfing the world wide web, connections are often times not already established but have to be set up before making the actual request. When serving a web page that includes individualized content, this can be a problem that should be accelerated.

To simulate this use case, an experiment similar to the one answering the first question was made, but with slightly changed parameters. Instead of probing the connection very frequently, requests were made every 30 seconds and the requested payload had a size of 100 kB.

Evaluating this measurement in section 5.4 revealed that there is a great benefit for overlay network routing for all node groups, even the ones that are geographically close and did not benefit in the previous experiment.

When measuring the complete download time, results were up to 40 % better using the overlay network, or around 80 ms lower in absolute values. Removing the time needed to perform the initial TCP handshake reduced the overlay network's advantage to ca. 15 %, but still is a surprisingly good result given the unevenly distributed nodes.

Because this measurement did not absolutely exclude any other form of optimization next to rerouting the traffic, it is possible that other factors used by the Akamai Overlay Network and its SureRoute service improved the download times measured during the experiment.

This shows what great impact from using acceleration techniques on download times can be, even without statically caching the content on distributed servers.

How similar are the transport routes that benefit the most from Overlay-Network-Acceleration?

To answer this question, case studies comparing different characteristic routes have been made and are described in section 5.5.

The analysis of three different routes showed several interesting aspects:

- In situations where the default Internet route is already optimal, like for most of the nodes within the same country as the web server for example, overlay routing adds a few milliseconds overhead. This is not because it tries to find an alternative route at any cost, but because there is just an extra amount of processing required, and of course a few extra hops to reach the Akamai server in the first place.
- There may be test nodes not too far away that show surprisingly bad plain Internet round trip times, because their route takes unnecessary hops that increase both physical transport way and the number of hops on the route. Overlay networks can in these cases find better routes by using completely different paths over different networks and hops in different cities compared to the original route.
- When measuring round trip times from locations with a great distance to the web server, it may be sufficient to choose a different network operator but essentially using the same path nevertheless.

The case study showed that there are different scenarios where an overlay network's routes can either be exactly the same if there is no better route available, which means that there is no benefit to be gained from pure overlay routing for those clients.

In another case, the routes could be completely different from the default Internet route where only the first and last few hops are shared between both routes. For clients with highly different routes, a benefit in round trip times is definitely expected, otherwise the overlay network would not have chosen such a different path.

The last case study showed that converse correlation between differing paths and increased performance does not need to be true: Even routes that take the same physical path can be faster if the hops are on different networks and therefore use different hardware that has an advantage of being faster or less used.

Finally, it has been showed that a overlay network accelerated route with 32 hops can still be faster than a plain Internet route with only 15 hops, which is a truly astonishing result and proof of overlay network's capabilities.

Bibliography

[1] C. A. Ellis and S. J. Gibbs, "Concurrency control in groupware systems," in *Proceedings of the 1989 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD '89. New York, NY, USA: ACM, 1989, pp. 399–407. [Online]. Available: http://doi.acm.org.eaccess.ub.tum.de/10.1145/67544.66963

[2] P. Baake and T. Wichmann, "On the economics of internet peering," *NETNOMICS*, vol. 1, no. 1, pp. 89–105, 1999. [Online]. Available: http://dx.doi.org/10.1023/A%3A1011449721395

[3] Akamai Technologies, Inc., "State of the internet q1 2015," https://www.stateoftheinternet.com/resources-connectivity-2015-q1-state-of-the-internet-report.html, 2015, [Online; accessed June 5 2015].

[4] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An analysis of internet content delivery systems," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 315–327, Dec. 2002. [Online]. Available: http://doi.acm.org/10.1145/844128.844158

[5] E. Nygren, R. K. Sitaraman, and J. Sun, "The akamai network: A platform for high-performance internet applications," *SIGOPS Oper. Syst. Rev.*, vol. 44, no. 3, pp. 2–19, Aug. 2010. [Online]. Available: http://doi.acm.org.eaccess.ub.tum.de/10.1145/1842733.1842736

[6] P. Manils, C. Abdelberri, S. Le Blond, M. A. Kaafar, C. Castelluccia, A. Legout, and W. Dabbous, "Compromising Tor Anonymity Exploiting P2P Information Leakage," *ArXiv e-prints*, Apr. 2010.

[7] Akamai Technologies, Inc., "Facts & Figures," http://www.akamai.com/html/about/facts_figures.html, 2015, [Online; accessed April 12 2015].

[8] ——, "TCP Optimizations," https://developer.akamai.com/stuff/Optimization/TCP_Optimizations.html, 2015, [Online; accessed May 27 2015].

[9] ——, "Akamai feature SureRoute," http://www.akamai.com/dl/feature_sheets/fs_edgesuite_sureroute.pdf, 2003, [Online; accessed April 12 2015].

[10] D. G. Andersen, A. C. Snoeren, and H. Balakrishnan, "Best-path vs. multi-path overlay routing," in *Proceedings of the 3rd ACM SIGCOMM Conference on Internet Measurement*, ser. IMC '03. New York, NY, USA: ACM, 2003, pp. 91–100. [Online]. Available: http://doi.acm.org.eaccess.ub.tum.de/10.1145/948205.948218

[11] Y. Chen, D. Bindel, H. Song, and R. H. Katz, "An algebraic approach to practical and scalable overlay network monitoring," in *Proceedings of the 2004 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '04. New York, NY, USA: ACM, 2004, pp. 55–66. [Online]. Available: http://doi.acm.org.eaccess.ub.tum.de/10.1145/1015467.1015475

[12] Y. Ren, Y. Qiao, X.-s. Qiu, and S.-a. Wu, "Scalable deterministic end-to-end probing and analytical method for overlay network monitoring," in *Proceedings of the 7th International Conference on Network and Services Management*, ser. CNSM '11. Laxenburg, Austria, Austria: International Federation for Information Processing, 2011, pp. 460–464. [Online]. Available: http://dl.acm.org.eaccess.ub.tum.de/citation.cfm?id=2147671.2147756

[13] R. Cohen and D. Raz, "Cost-effective resource allocation of overlay routing relay nodes," *IEEE/ACM Trans. Netw.*, vol. 22, no. 2, pp. 636–646, Apr. 2014. [Online]. Available: http://dx.doi.org.eaccess.ub.tum.de/10.1109/TNET.2013.2260867

[14] M. Shahzad and A. X. Liu, "Noise can help: Accurate and efficient per-flow latency measurement without packet probing and time stamping," in *The 2014 ACM International Conference on Measurement and Modeling of Computer Systems*, ser. SIGMETRICS '14. New York, NY, USA: ACM, 2014, pp. 207–219. [Online]. Available: http://doi.acm.org.eaccess.ub.tum.de/10.1145/2591971.2591988

[15] E. Halepovic, J. Pang, and O. Spatscheck, "Can you get me now?: Estimating the time-to-first-byte of http transactions with passive measurements," in *Proceedings of the 2012 ACM Conference on Internet Measurement Conference*, ser. IMC '12. New York, NY, USA: ACM, 2012, pp. 115–122. [Online]. Available: http://doi.acm.org.eaccess.ub.tum.de/10.1145/2398776.2398789

[16] V. Paxson, "End-to-end routing behavior in the internet," *SIGCOMM Comput. Commun. Rev.*, vol. 36, no. 5, pp. 41–56, Oct. 2006. [Online]. Available: http://doi.acm.org.eaccess.ub.tum.de/10.1145/1163593.1163602

[17] H. V. Madhyastha, E. Katz-Bassett, T. Anderson, A. Krishnamurthy, and A. Venkataramani, "iplane nano: Path prediction for peer-to-peer applications," in *Proceedings of the 6th USENIX Symposium on Networked Systems Design and Implementation*, ser. NSDI'09. Berkeley, CA, USA: USENIX Association, 2009, pp. 137–152. [Online]. Available: http://iplane.cs.washington.edu/nsdi09.pdf