University of Zurich^UZH

BACHELOR THESIS – Communication Systems Group, Prof. Dr. Burkhard Stiller

Data Aggregation and Visualization for the Torrent Measurement Software Kraken

Sebastian Schrepfer
Oberwil-Lieli, Switzerland
Student ID: 10-737-567

Supervisor: Andri Lareida, Dr. Thomas Bocek
Date of Submission: December 11th, 2014

University of Zurich
Department of Informatics (IFI)
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland

ifi

Bachelor Thesis
Communication Systems Group (CSG)
Department of Informatics (IFI)
University of Zurich
Binzmühlestrasse 14, CH-8050 Zürich, Switzerland
URL: http://www.csg.uzh.ch/

Abstract

The share of BitTorrent in the global Internet traffic is, despite the rise of movie on-demand platforms, still very high. As a contribution to the research in this field, the Kraken measurement study gathers metadata from peers which are downloading movies in BitTorrent networks. The large bulk of data is solely stored in a database, which makes it difficult to interpret. In the scope of this Bachelor Thesis, a web interface has been developed, which derives meaningful information from the raw data and allows online access for fellow researchers. The Kraken Web Interface automatically aggregates the data and presents the obtained information in cartographic and in time-based visualizations. The visualizations focus on the integrity of the supplied information, the usability of the application and an appealing design.

Der Anteil von BitTorrent im weltweiten Internettraffic ist trotz des Wachstums von Video-on-Demand Angeboten noch immer sehr hoch. Die Software der Studie Kraken beschäftigt sich deshalb mit Filmdownloads aus BitTorrent-Netzwerken und zeichnet Metadaten anderer Teilnehmer auf. Die gewonnene Datenmenge ist sehr gross und nur in einer Datenbank hinterlegt, was die Interpretation dieser Daten erheblich erschwert. Im Rahmen dieser Bachelorarbeit wurde eine Weboberfläche als Schnittstelle entwickelt, um aus den Rohdaten aussagekräftige Information zu erhalten und online zu veröffentlichen. Das Kraken Web Interface aggregiert die Daten und präsentiert die gewonnen Information in kartographischen und zeitbasierten Visualisationen. Die Darstellung der Informationen orientiert sich an der Korrektheit der dargestellten Informationen sowie der Benutzerfreundlichkeit und dem Design der Webapplikation.

Acknowledgments

First and foremost, I would like to express my sincere gratitude to my supervisor Andri Lareida, who supported me throughout the course of this thesis. He offered his continuous advice and encouragement. Without him, this thesis and the work on the Kraken Web Interface would not have been possible. I thank him for the systematic guidance and the great effort he put into supporting me.

I would also like to thank the 19 persons, who participated in the questionnaire about the usability evaluation of the website. A special thanks goes to the participants who provided an extended feedback with an explanation to their evaluation and directions for possible improvements to the software.

My sincere thanks also goes to Annika Sinkwitz and Christina Sinkwitz for helping me improve the english grammar and syntax of this thesis.

Contents

Chapter 1

Introduction

The share of BitTorrent in Internet traffic is worldwide very high. The upstream traffic of BitTorrent is on top of the ranking of the peak period applications worldwide [54]. Its upstream share lies between 19.8% in Latin America and 45.7% in Asia and Pacific. With this said, it is more than reasonable to study the underlying network of BitTorrent.

Kraken is a large scale BitTorrent measurement software which gathers data of the users of the BitTorrent network. It sends announce requests for movies and collects the returning data. This data mainly contains IP addresses of other participants of the particular movie download. The functionality of Kraken ceases with successfully storing of the data into a database. The raw data is neither aggregated nor interpreted by the existing Kraken Core software.

This Bachelor Thesis takes up the work of Kraken Core and aims to provide a tool for the continuous interpretation and visualization of the aggregated data in the form of a web application. The front-end of the application mainly depicts the peer localization on various maps. Those maps are filled with data of a general observation of all of the tracked movies together, but can also be filtered to show data of a single movie on a selected observation day. Furthermore, the Kraken Web Interface displays the current status of the Kraken Core software in order to give the administrator a quick overview about the current Kraken infrastructure.

The aggregation and visualization of the gathered data leads to new insights. It helps to further examine and research the BitTorrent network by displaying the relevant data and providing a visual interface.

1.1 Motivation

When developing an application, the visualization and usability is often put in the rear. The main goal of the application is designed and implemented, while the visual and interactive part remains disregarded. With Kraken, only the core application had already been built, but the visual part was still missing.

It has been proven that the perceived usability has a direct and positive relationship on the degree of the consumer's trust and satisfaction [23]. Hence, it is obvious that a software needs to be provided with a usable, interactive user interface. Building a front-end web interface was therefore a strong reason for working on the Kraken Web Interface.

The Kraken Web Interface has been created with both newer technologies like D3 and Sass, but also with more seasoned technologies like the Java socket connections. Creating a web project with those unfamiliar technologies and learning how to apply them in a web project was challenging, but also very interesting. The technological challenges were another reason for working on the Kraken Web Interface.

As mentioned before, BitTorrent has a great significance in global Internet traffic. Examining the network and analyzing its data could lead to research with major impact on the architecture of the BitTorrent network. Having the opportunity to make a contribution to this research feels very exciting to me.

1.2 Description of Work

This Bachelor Thesis visualizes the obtained information of raw data provided by the Kraken Core Software. The work contains the design, architectural planning and implementation of the Kraken Web Interface.

The Kraken Web Interface is designed to be user-friendly and interactive. It is completely mobile optimized and offers dynamic charts. The charts display by choice very specific or generic data. Additionally, the Kraken Web Interface provides an overview of the current system status of the Kraken Core software.

The application can be roughly divided into a back-end and a front-end part. The back-end part contains the aggregation from the data, which has been collected by the Kraken Core software. It removes duplicates and converts the raw data into a state in which it can be accessed and processed by the front-end. The web server, which is a part of the back-end, then acts as a proxy between the aggregated data and the front-end. There has also been established a connection to the running Kraken Core software in order to obtain the latest system status. In the front-end part of the application, the data is being received and visualized in various formats. A map displays the information in a movie-specific or in a general way. A time chart displays the progression of the number of peers of each observed torrent over time. Furthermore, the system status is presented in a table.

Eventually, a usability evaluation questionnaire was conducted. The participants were asked to familiarize with the software, whereupon they were asked to answer some contextual questions. In the next step, the default questions of the System Usability Scale (SUS) were asked, followed by some optional demographic questions. The analysis of the questionnaire led to interesting results.

1.3 Thesis Outline

This thesis begins by describing the technologies used in the front-end and in the back-end of the Kraken Web Interface. Chapter 2 also describes the related work, such as BitTorrent, Kraken Core or the SUS scale. Since this work has a strong focus on the development of a technical application, the engaged technologies will be described in detail.

In Chapter 3, the design of the Kraken Web Interface will be discussed. This includes the concepts of all visible parts throughout the web front-end, in particular the three front-end modules ”Generic information”, ”Specific information” and ”System Status”. The functionality of these modules and their components will be described in detail. One section will be devoted to the navigation on the website. Because the web interface is fully mobile optimized, there will be a separate section on the mobile optimization.

Subsequently, Chapter 4 explains the architecture of the Kraken Web Interface and its connection to the Kraken Core software. There will be described how the Kraken Web Interface connects with its own database to the database of the Kraken Core and aggregates the data on a regular basis. Furthermore, there has been established a socket connection between the two, for transmitting the current system status. This chapter also considers the details of the data aggregation.

The beginning of the chapter on the implementation covers the JavaScript Module Pattern. The front-end JavaScript code has been structured alongside this pattern. Chapter 5 continues with a section on the build process which includes comments on how a Maven build process evolved with the CSS precompiler Sass and its framework Compass. Furthermore, there will be provided a detailed explanation of the conversion of the raw map data to the effective map in the front-end. In the end of the chapter, the Java back-end and its socket connection to the Kraken Core will be described in detail.

Chapter 6 discusses the usability evaluation. It begins with a section on the Scenario & Setup of the questionnaire, continues with the results and finishes with their interpretation and analysis.

In the final chapter, the thesis will be summarized. In addition, the results of the thesis will be discussed. The thesis ends with a section on potential future work.

Chapter 2

Related Work & Technologies

This chapter discusses related work and the technologies used in the development of the Kraken Web Interface. First, an overview of BitTorrent will be given, then the Kraken Core Software will be described. Finally an overview of the front-end technologies will be described, followed by a section of the back-end technologies used to implement the Kraken Web Interface.

2.1 BitTorrent

BitTorrent (BT) is a peer-to-peer file sharing network protocol, as specified on the official website [19]. The network consists of peers and trackers, the latter helps the peers to find each other. A peer downloads splitted parts, chunks, of the requested file from a seeder. The seeder does not have to be in possession of the full file, only the requested part of it, in order to provide it to other peers.

A user can join a torrent network with a small torrent file, which contains metadata of the desired file. This metadata contains information about the file name, length, hashing information and the url of the tracker [18] [19].

A centralized server, called tracker, is responsible for helping downloaders find each other [18]. A peer connects to the tracker to receive a list of other peers' ip addresses and port numbers. It then connects to the received ip addresses simultaneously and downloads the different pieces of the file from different peers [53]. The tracker therefore acts as a distribution server of ip addresses and port numbers.

2.2 Kraken Core Software

Kraken is a torrent measurement system which investigates the behavior of BT users when downloading movies from a BitTorrent network. It consists of multiple slaves and one master. The connection between master and slave is established with a TCP socket connection.

The master connects in a predefined time interval to the torrent plattform Kickass Torrents [46]. The platform provides an RSS feed with the newest movie torrents available. This feed is parsed by the master which will then download all the current torrents from that feed. The downloaded torrents are forwarded to all slaves which are currently registered to the master.

The slaves connect to each tracker of a torrent file repeatedly. The list of ip addresses of peers, which are currently downloading the movie, is returned and forwarded to the master. The master stores this dataset into the database. This procedure is being repeated until a torrent is marked as inactive. An torrent is defined as inactive when the number of connected peers have dropped below a predefined threshold.

2.3 Front-End Technologies

In this chapter, the technologies used to create the web front-end of the Kraken Web Interface will be described. Front-end in this context consists of the web-based technologies. The back-end is separated from the front-end by the JSON interface, which provides data from the back-end to the the front-end, where the data will be processed.

This chapter starts by describing the styling of a website, particularly CSS, Sass/Compass. It continues with the JavaScript frameworks and libraries jQuery, D3.js and C3.js. Eventually there will be provided a paragraph about the compression of JavaScript and CSS.

2.3.1 Cascading Style Sheets

Cascading style sheets (CSS) [61] is the standard language when formatting web pages. While the HTML of a website is only useful for structuring a website, CSS uses selectors to identify a HTML element. Every kind of style (fonts, colors, margins, etc.) can then be added to a HTML element with CSS.

The third version of CSS supports individual fonts, rounded edges on elements and also background gradients. As it is not yet implemented in older versions of some browsers (especially in Internet Explorer), the websites which are optimized for CSS3 may look different in certain browsers. There exist a few frameworks for CSS3. Being one of them, Normalize.css is useful to make browsers render the HTML elements more consistently and in line with modern standards [37]. Instead of just resetting all styles for cross-browser equality, when starting a new project, it provides default styling standards.

2.3.2 Sass/Compass

Sass [15] is a CSS pre-compiler software which was developed in Ruby [48]. Sass allows to use functions, variables, and nesting of css rules which facilitate the organization of the CSS code. Compass [21] runs on Sass and provides plenty of useful predefined functions. One of the strengths of compass is its uncomplicated way of handling browser prefixes. To browser suppliers, the so-called vendor-prefixes provide a way of introducing a new CSS functionality. For all major browsers, there exists a vendor-prefix for the CSS property *user-select*. Therefore, it is necessary to use four properties instead of just one: *-webkit-use-select*, *-moz-user-select*, *-ms-user-select* and user-select. Compass, however, provides one function *user-select* which then compiles to the vendor-specific css rules. Together with Sass, it is an indispensable tool in the modern web development.

As Sass is written in Ruby, it cannot regularly be integrated into a maven process. Fortunately though, a solution exists for this problem: jRuby [5]. jRuby runs as a standard java library but caters a full running instance of Ruby. After performing a recompilation of ruby together with Sass/Compass modules, the library will be able to compile the sass code into css rules, which can be run in a java process and therefore also during a Maven [4] build process. The downside of this procedure consists in the cumbersome way of updating the Sass/Compass library. Consult Chapter 5.2.1 for further information.

2.3.3 jQuery

jQuery is a standard JavaScript library which presents a simple interface to the DOM elements of a website. It eliminates the different ways in which browsers implement the JavaScript engine by simply introducing one function, which then handles the browser differences. It also provides many functions for looping through different DOM elements. jQuery supports the extensibility of the framework through plugins. The following three jQuery plugins were used in the Kraken Web Interface.

jQuery Template [13] allows to store a template in the form of an HTML snippet in a script tag with content type text/html. The variables, which are embedded in the data attributes of the HTML DOM elements, will be scanned in every *loadTemplate()* call and prefilled with the data given to the plugin. It is also possible to add formatting functions to preformat the given values.

jQuery Tablesorter [38] contributes a sorting functionality to an HTML table. It automatically adds ClickListeners to the table header row and provides many configuration possibilities - including the specifying of the css arrow classes. The paging plugin of the tablesorter also bringsthe possibility of splitting the table content into multiple pages, despite the sorting feature.

jQuery Pickadate [1] is a mobile optimized date picker. Its visual interface consists of a calendar where one can select a date. It can be configured in a certain way, that only the days are selectable.

2.3.4 D3 and Related Technologies (Frameworks)

Data Driven Documents, also called D3.js, is a JavaScript framework which is used to visualize data in the web browser. It manages to create graphs which can be made interactive. To visualize the graphs, D3.js relies heavily on Scalable Vector Graphics (SVG) standard.

SVG is a standard for displaying two-dimensional vector graphics. Due to the graphics being defined in a vector and not in pixels, the images are stored in XML text files. It is also possible to directly include them in the HTML markup. The graphics can be animated with JavaScript.

D3.js can be used to select and modify DOM and SVG elements, but it also provides a large set of functions to facilitate the visualization for the software engineer. It makes use of SVG together with the regular browser standards HTML and CSS and provides low level functionality for individual graphs, but also high level functionality in maps and some other areas. D3.js offers an API for the creation of maps. The map coordinates, which are specified in JSON, can be passed on to D3.js, together with a map projection, in order to eventually create the map.

In order to create a map with D3, it is necessary to provide the map data in the JSON format. The shape file, however, has evolved to a leading map data transfer standard. It is mainly being used in geographic information system software (GIS) [59]. For that reason, a conversion between a shape file and JSON is inevitable.

GeoJSON [14] is an open standard for encoding geographical features in the JavaScript Object Notation JSON. It consists of coordinates and metadata about a map entity (for example a country). GeoJSON can be directly passed on to D3.js in order to visualize a map according to the map data.

TopoJSON [9] extends and optimizes GeoJSON. It reduces redundancy by combining and storing shared borders of countries only once. By removing the redundancy, the file size can be largely decreased. In an example provided on the website of TopoJSON [11], the file size of a GeoJSON file with all the US counties could be reduced by 80% when using TopoJSON instead. TopoJSON is available as a command line tool to encode map data from GeoJSON or shape files to TopoJSON, and as a JavaScript extension of D3.js for interpreting the encoded map data.

C3.js [56] is a chart library which relies on D3.js. It provides customizable charts, mainly standard line and area charts. In contrary to D3.js, most of the functionality is not programmable, but configurable. As it was first released in May, 2014, most of the functionality was still undocumented at the time when the Kraken Web Interface was developed.

2.3.5 Further Front-End Technologies

Moment.js [62] is a framework for parsing, manipulating and displaying times and dates in JavaScript. It is useful for calculating or simply for displaying dates from another format on a website. It runs in browsers, but also in Node.js.

The **YUI compressor** [63] minifies JavaScript and CSS files to a minimal byte footprint while preserving the operational qualities of the code. In JavaScript, it removes all unnecessary whitespace characters and renames the local variables and functions names to one-character names (in some cases up to three characters). In CSS, it uses multiple regular expressions in order to remove the unnecessary whitespace.

The YUI compressor is written in Java and can therefore easily be integrated into a Maven build process.

2.4 Back-End Technologies

This chapter describes the technologies used in the Java back-end of the Kraken Web Interface. It starts with the web server Apache Tomcat and continues with the MySQL database and the Java Servlet Pages, which are necessary for the supply of HTML fragments. In the end, the build process tools Apache Maven and Apache Ant are described.

Apache Tomcat
Apache Tomcat is an open source web server which implements the specifications for Java Servlets. It allows the use of Java in servlet/jsp based web applications.

Java Servlet Pages (JSP)
Java Servlet Pages (JSP) are HTML Pages with additional Tags. For example, another JSP file can be included into the page by usage of the tag <jsp:include/>. This allows greater flexibility in coding reusable components. It also supports tag libraries, which are supplied as frameworks. With the *c taglib*, for example, one can integrate loops and conditions with tags directly into the JSP files. It also supports variables which can be defined by the servlet.

MySQL Database
MySQL is a database which supports scheduled executions of SQL queries. The queries can be stored in the database, similar to views. MySQL also supports the federated engine, which is able to create views of tables of another MySQL database over network.

Apache Maven
Apache Maven is a software project management and comprehension tool [4]. Maven is used to manage project builds by one central file called pom.xml.

Apache Ant
Apache Ant is a Java library which can be employed to run tasks [2]. Not unlike

Apache Maven, it is used to build Java applications. Ant also manages to run file based operations and java applications. Therefore, it is suitable for running a Sass compilation and a YUI compression in a script.

2.5 System Usability Scale (SUS)

The System Usability Scale (SUS) was developed by Brooke (1996) [12] as a "quick-and-dirty" usability scale. It is a simple ten-item scale, giving a global view of subjective assessments of usability [12]. SUS is a Likert scale, where the respondent indicates the degree of agreement or disagreement with their statement on a five-point scale.

The SUS is technology agnostic and applicable for a wide range of interface technologies [7]. It is easy to use for both study participants and administrators, it provides a single score that is easily understood by a wide range of people, and it is non-proprietary [7].

Brooke described the calculation of the SUS value in his paper in the following way: The result is calculated through the answers of the single questions. The scale response of every second question (1, 3, 5, 7, 9) is subtracted by one. For the scale response of all the other questions (2, 4, 6, 8, 10) the value is subtracted from 5. Multiplying the sum of the scores by 2.5 leads to the overall score of the SUS, which is a value in the range of 0 to 100. Higher scores indicate better usability.

Bangor et al. [7] described the score of SUS as highly reliable. They also discovered that there is a significant, but not very strong correlation between the age of the respondent and the SUS score, with a higher age having a negative impact to the SUS score. However, according to Bangor et al., there is no significant difference between the mean SUS scores of women or men.

2.5.1 Interpretation of the SUS Score

As a standard rule of thumb, a typical grading scale called university grading scale [7] has evolved. A SUS score from 90 to 100 related to an A, a score from 80 to 89 to a B and so on. Despite this concept being very handy, it has not yet been validated [7].

In a subsequent study, Bangor et al. (2009) [6] added an 11^{th} question to the SUS survey: "Overall, I would rate the user-friendliness of this product as:" with 7 possible answers. The SUS score of 212 surveys was then mapped to those adjectives, which produced a mean SUS score of every single adjective. Table 2.1 displays the SUS scores mapped to the corresponding adjective. With exception of the adjectives "worst imaginable" and "awful", all of the adjectives were significantly different.

Adjectives	Mean SUS Score
Worst imaginable	12.5
Awful	20.3
Poor	35.7
OK	50.9
Good	71.4
Excellent	85.5
Best Imaginable	90.9

Table 2.1: This table represents the adjectives mapped to the mean SUS score.

Chapter 3

Design

The Kraken Web Interface consists of three disparate modules *Generic information*, *Specific information* and *Current System Status*. The **Current System Status** displays the current state of the Kraken Core software. As **Generic information**, a project description page provides information about the Kraken Web Interface and the Kraken Core software. It is also equipped with a world map containing all collected data from one specific day. As **Specific information**, a page with a filterable torrent list displays all observed movie torrents from one specific day. Furthermore, there is meta information about every torrent in the list displayed. Each torrent can be selected for receiving detailed information. A Torrent Details Page with a time graph and a specific map for the torrent on one day will open. The design of the modules is depicted in Figure 3.1.

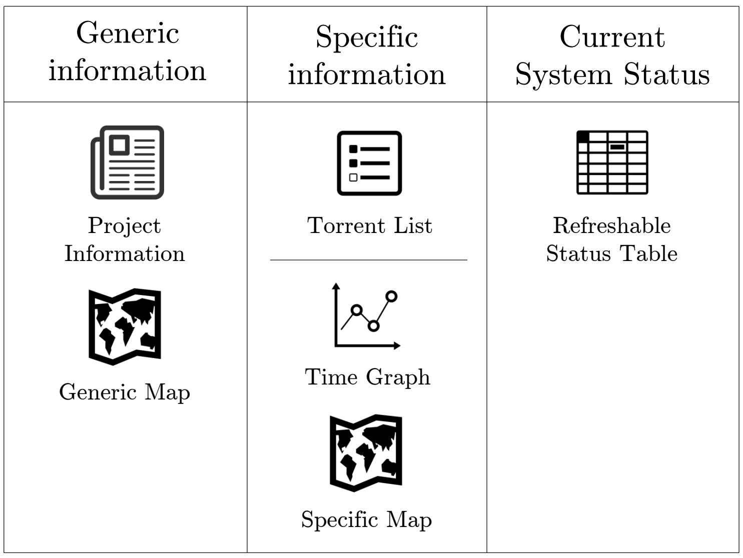


Figure 3.1: The Kraken Web Interface Modules: Generic information, specific information and the current system status. Image Icons: [20, 36, 25, 30, 33]

3.1 Generic Information

The generic information consists of a page with a project description. The description gives an overview of the Kraken Core software and in more detail the Kraken Web Interface. This page contains a map with generic torrent movie data. The data is an insight into the torrent observation from one specific day. It displays the distribution of torrent peers over to world: from the most active countries to the non-participating countries. In Figure 3.2 the project information and the generic map will be depicted.

The project description page provides an entrance to the Kraken project. The information about the project explains the reasons behind the Kraken Core software and the Web Interface. The user is not overwhelmed by too much information, due to the torrent specific pages being on separate pages. Nevertheless, they still get an insight into the project with the description and the generic map.

3.2 Specific Information

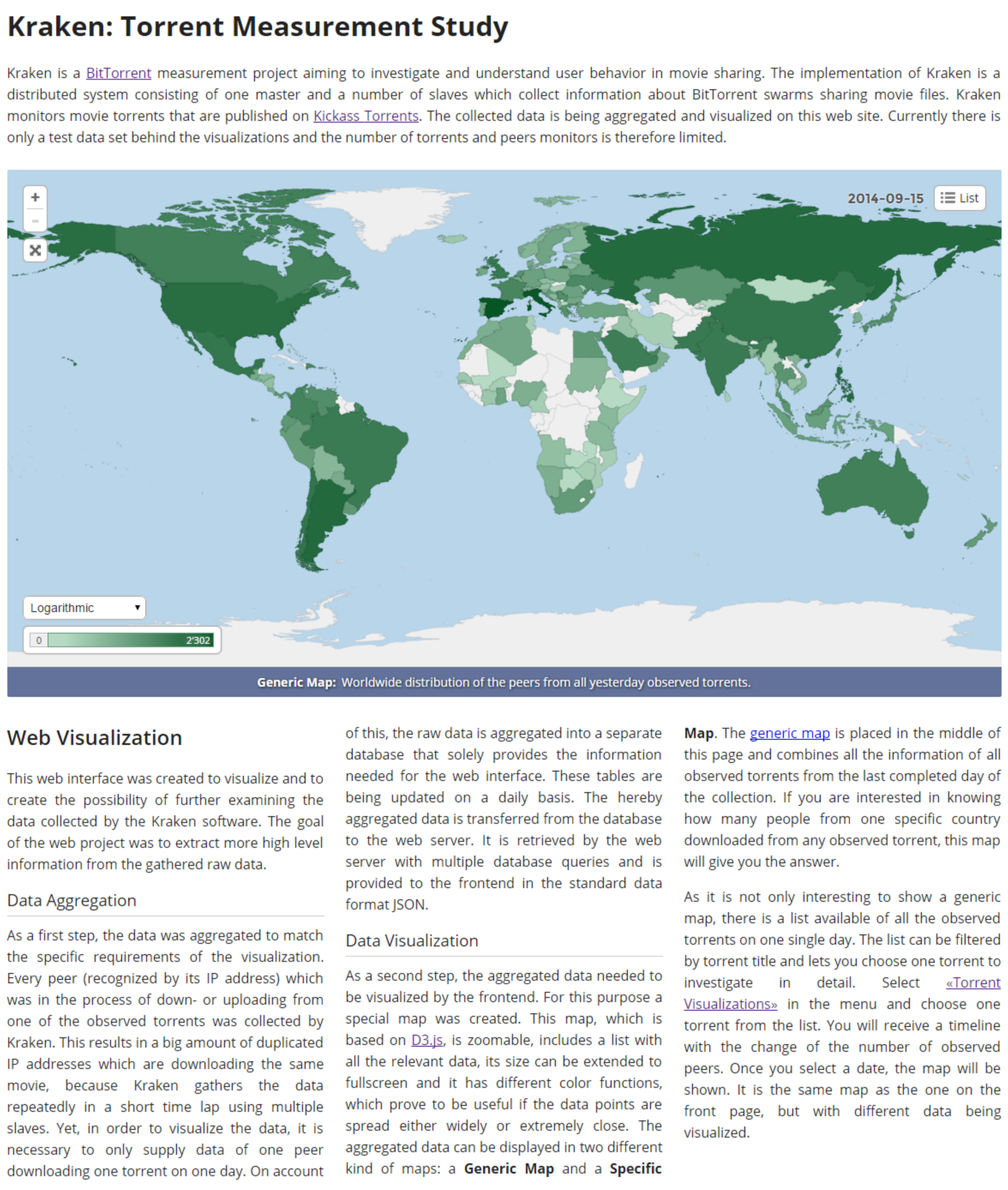
The specific information consists of two separate pages. The first page provides a filterable page with all observed torrents from one day. The user can select one torrent and will receive more detailed information about this specific torrent on the next page.

3.2.1 Torrent List

The torrent list, depicted in Figure 3.3, provides an overview of torrents, which were observed on one single day. Per default, the most up to date day with stored data will be preselected. The preselected day regularly equals the day prior to the current calendar date, because the data aggregation takes only place once a day at midnight.

One row displays metadata of a movie torrent: The movie title, the publish date and the file size are presented. It also displays the aggregated data: observed peers and maximal swarm size on the selected day. A user can select one torrent and will be redirected to the details page.

Users can click on the date, which will make a calendar with all selectable days appear. This calendar is depicted in Figure 3.5. It is also possible to switch to the next or to the previous day without opening the calendar. The results are filterable by keyword. The results are filterable by keyword. In Figure 3.4 a screenshot of the keyword filtered list is depicted. Only one movie is found when entering the keyword ”mamula”. The date and the keyword, if stated, are deposited in the URL. Thus, users can bookmark the url and the same data will be presented on their return.



Kraken: Torrent Measurement Study

Kraken is a BitTorrent measurement project aiming to investigate and understand user behavior in movie sharing. The implementation of Kraken is a distributed system consisting of one master and a number of slaves which collect information about BitTorrent swarms sharing movie files. Kraken monitors movie torrents that are published on Kickass Torrents. The collected data is being aggregated and visualized on this web site. Currently there is only a test data set behind the visualizations and the number of torrents and peers monitors is therefore limited.

Generic Map: Worldwide distribution of the peers from all yesterday observed torrents.

Web Visualization

This web interface was created to visualize and to create the possibility of further examining the data collected by the Kraken software. The goal of the web project was to extract more high level information from the gathered raw data.

Data Aggregation

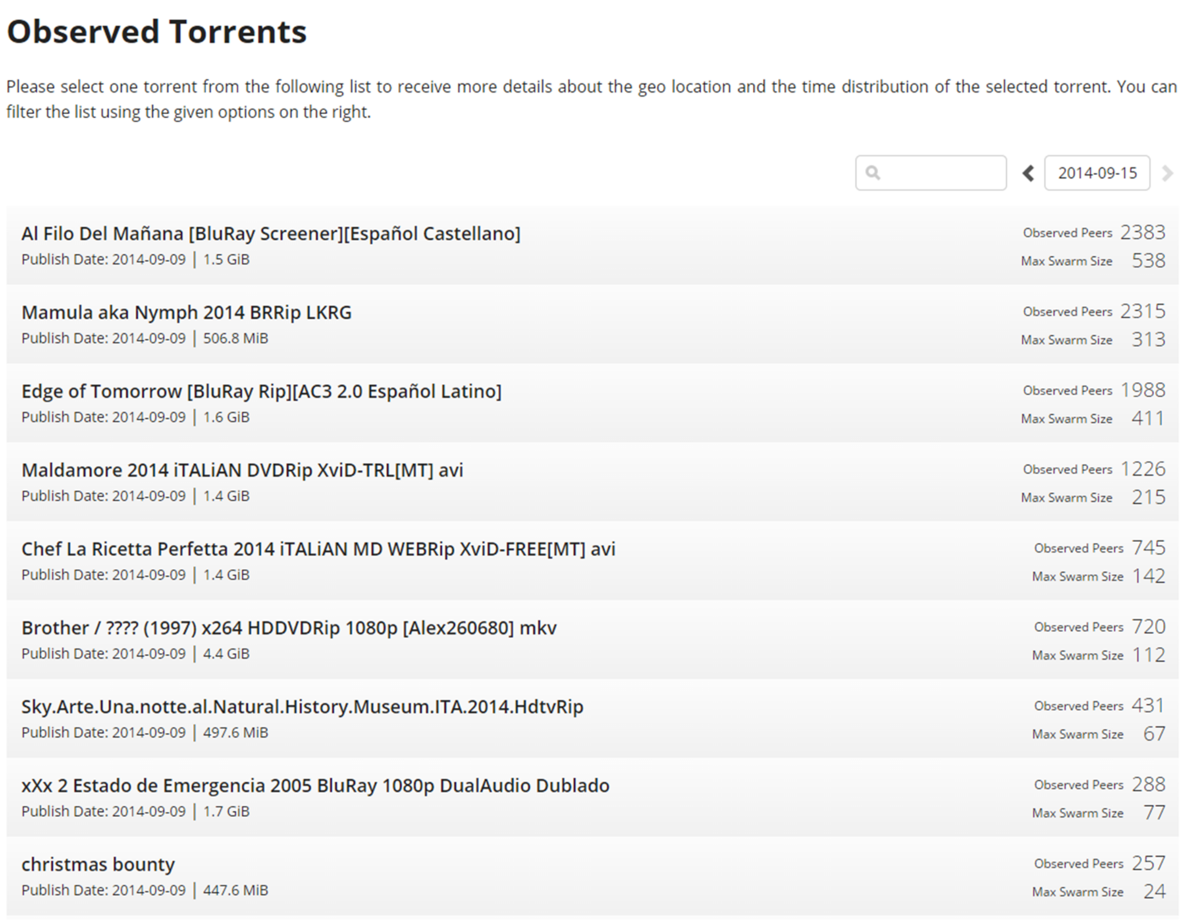
As a first step, the data was aggregated to match the specific requirements of the visualization. Every peer (recognized by its IP address) which was in the process of down- or uploading from one of the observed torrents was collected by Kraken. This results in a big amount of duplicated IP addresses which are downloading the same movie, because Kraken gathers the data repeatedly in a short time lap using multiple slaves. Yet, in order to visualize the data, it is necessary to only supply data of one peer downloading one torrent on one day. On account of this, the raw data is aggregated into a separate database that solely provides the information needed for the web interface. These tables are being updated on a daily basis. The hereby aggregated data is transferred from the database to the web server. It is retrieved by the web server with multiple database queries and is provided to the frontend in the standard data format JSON.

Data Visualization

As a second step, the aggregated data needed to be visualized by the frontend. For this purpose a special map was created. This map, which is based on D3.js, is zoomable, includes a list with all the relevant data, its size can be extended to fullscreen and it has different color functions, which prove to be useful if the data points are spread either widely or extremely close. The aggregated data can be displayed in two different kind of maps: a **Generic Map** and a **Specific Map**. The generic map is placed in the middle of this page and combines all the information of all observed torrents from the last completed day of the collection. If you are interested in knowing how many people from one specific country downloaded from any observed torrent, this map will give you the answer.

As it is not only interesting to show a generic map, there is a list available of all the observed torrents on one single day. The list can be filtered by torrent title and lets you choose one torrent to investigate in detail. Select «Torrent Visualizations» in the menu and choose one torrent from the list. You will receive a timeline with the change of the number of observed peers. Once you select a date, the map will be shown. It is the same map as the one on the front page, but with different data being visualized.

Figure 3.2: Project information and generic map as the entry page when visiting the Kraken Web Interface.



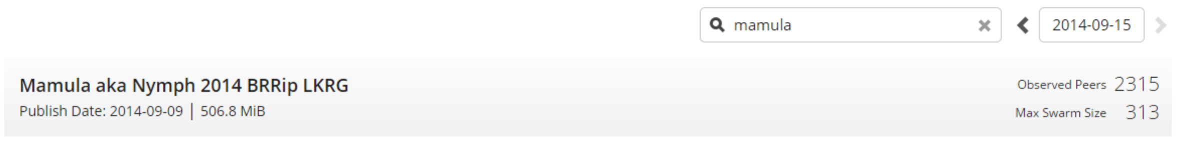
Observed Torrents

Please select one torrent from the following list to receive more details about the geo location and the time distribution of the selected torrent. You can filter the list using the given options on the right.

2014-09-15

Torrent	Publish Date	Size	Observed Peers	Max Swarm Size
Al Filo Del Mañana [BluRay Screener][Español Castellano]	2014-09-09	1.5 GiB	2383	538
Mamula aka Nymph 2014 BRRip LKRG	2014-09-09	506.8 MiB	2315	313
Edge of Tomorrow [BluRay Rip][AC3 2.0 Español Latino]	2014-09-09	1.6 GiB	1988	411
Maldamore 2014 iTALiAN DVDRip XviD-TRL[MT] avi	2014-09-09	1.4 GiB	1226	215
Chef La Ricetta Perfetta 2014 iTALiAN MD WEBRip XviD-FREE[MT] avi	2014-09-09	1.4 GiB	745	142
Brother / ???? (1997) x264 HDDVDRip 1080p [Alex260680] mkv	2014-09-09	4.4 GiB	720	112
Sky.Arte.Una.notte.al.Natural.History.Museum.ITA.2014.HdtvRip	2014-09-09	497.6 MiB	431	67
xXx 2 Estado de Emergencia 2005 BluRay 1080p DualAudio Dublado	2014-09-09	1.7 GiB	288	77
christmas bounty	2014-09-09	447.6 MiB	257	24

Figure 3.3: Torrent List with all torrents from one day. The list is filterable and the date can be switched.

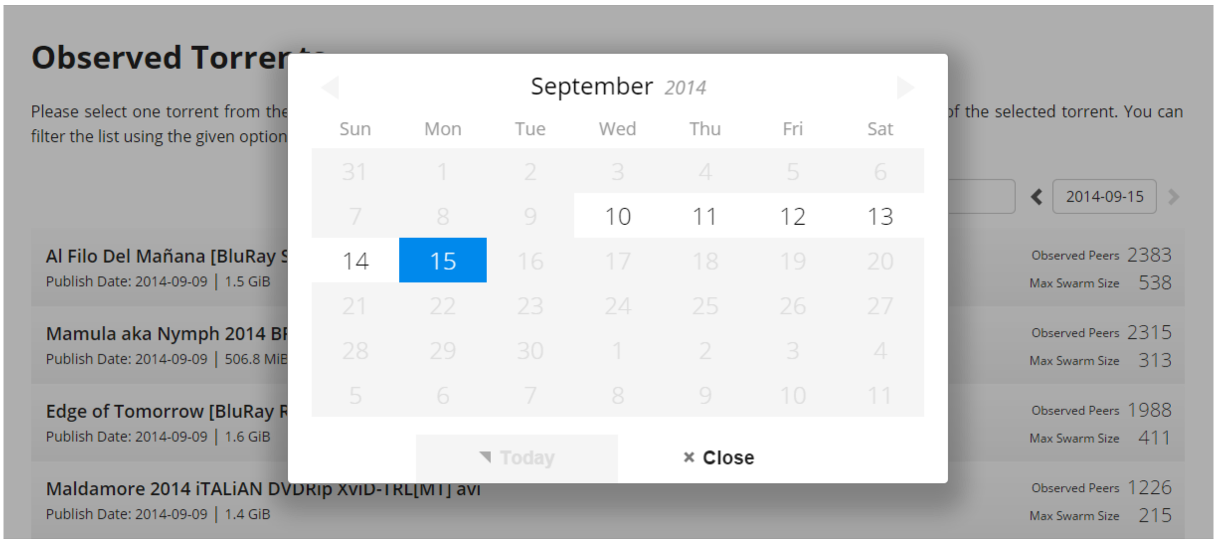


mamula

2014-09-15

Torrent	Publish Date	Size	Observed Peers	Max Swarm Size
Mamula aka Nymph 2014 BRRip LKRG	2014-09-09	506.8 MiB	2315	313

Figure 3.4: The torrent list is filterable. In this example, the word ”mamula” leads to one movie entry.


Observed Torrents

Please select one torrent from the … of the selected torrent. You can filter the list using the given option…

September 2014

Sun	Mon	Tue	Wed	Thu	Fri	Sat
31	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	1	2	3	4
5	6	7	8	9	10	11

Today × Close

2014-09-15

Al Filo Del Mañana [BluRay S…
Publish Date: 2014-09-09 | 1.5 GiB
Observed Peers 2383
Max Swarm Size 538

Mamula aka Nymph 2014 B…
Publish Date: 2014-09-09 | 506.8 MiB
Observed Peers 2315
Max Swarm Size 313

Edge of Tomorrow [BluRay R…
Publish Date: 2014-09-09 | 1.6 GiB
Observed Peers 1988
Max Swarm Size 411

Maldamore 2014 iTALiAN DVDRip XviD-TRL[MT].avi
Publish Date: 2014-09-09 | 1.4 GiB
Observed Peers 1226
Max Swarm Size 215

Figure 3.5: Selection of a date with observed torrents.

3.2.2 Observed Torrent Details

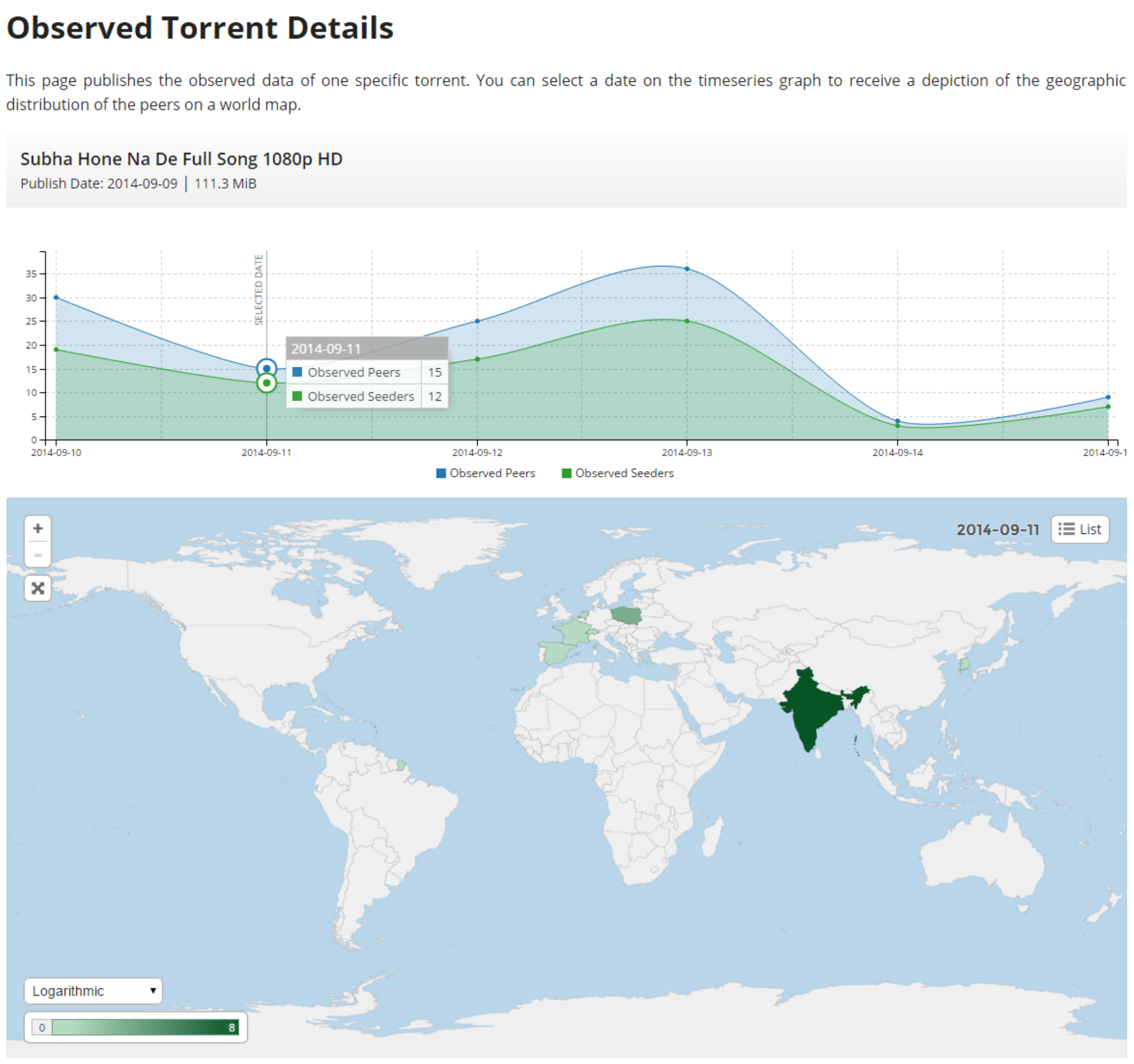
The observed torrent details page contains two sorts of graphs. The page is depicted in Figure 3.6. The first graph displays a timeline from the beginning of the observation of the torrent to the end. The end of the observation is either the final end of the observation, when a torrent has been marked as inactive, or just the date of the last data aggregation, hence the previous day. It displays the number of peers and the number of seeders on each date. By visualizing this data with a timeline graph, the user gains a quick impression of where the peaks, the ups and downs lie. This graph does not contain geographical information. It is the sum of the observation of all countries. One of the dates displayed on the graph can be selected to receive a specific map of the torrent on the selected date, which is the second graph.

After selecting a date on the graph, there will appear a map below the graph, which is based on the specific dataset. This map visualizes the data from the selected date and torrent. Another date can be selected, which will make the map refresh. A date can also be unselected, whereupon the map will disappear.

The selected date will be stored in the URL. The user can then bookmark the address, which will lead him back on the same page.

3.3 The Map

In order to understand the distribution of the peers in the BitTorrent network, it is highly interesting to comprehend from which geographic locations the movies are being downloaded. When visualizing the origin of a peer, a map is the obvious form of depiction. The map displays the number of observed peers, the maximal swarm size per country and the share in percentage, visualized per day.



Observed Torrent Details

This page publishes the observed data of one specific torrent. You can select a date on the timeseries graph to receive a depiction of the geographic distribution of the peers on a world map.

Subha Hone Na De Full Song 1080p HD

Publish Date: 2014-09-09 | 111.3 MiB

Figure 3.6: The torrent details page displays a time chart and a specific map.

The countries' shares were visualized in the map by using different color shapes for each country. The darker a country is displayed, the more people have downloaded the movie from within this country.

3.3.1 Color Functions

Due to the data changing every day, it is possible that on one day, the majority of peers are downloading from a single country, but on the next day the shares are equilibrated. With regard to visualizing those kind of contortions, it is necessary to choose from multiple color functions, which calculate the values between the maximum and minimum color.

There is a set of different color mapping functions available. The functions were selected from the basic set of elementary mathematical functions. They can be extended by new functions easily. The linear color function is the most straightforward example. It calculates the percentage value of one country in dependence of the maximum value of the countries. This value is then transformed directly into a color between the maximum and the minimum color.

With the logarithmic scale, as another example, the number of observed peers is first passed on as an argument to the logarithmic function. Following that, the percentage value in dependence of the logarithmic scaled maximum of one of the observed peers is transformed into a color, in a similar way to the process with the linear function. This example leads to a different distribution of the colors. Higher values are less important than in the linear scale, lower values a fortiori. The logarithmic scale was chosen as the standard color function for the map. It does not react as strong to distortion values as the linear functions, what leads to a general improvement of the color distribution. In Figure 3.7 the linear color function is used to render the map. In Figure 3.8 the logarithmic one was used to display the same dataset as in the linear one. The comparison of those two maps should give an impression, about the usefulness of various color functions.

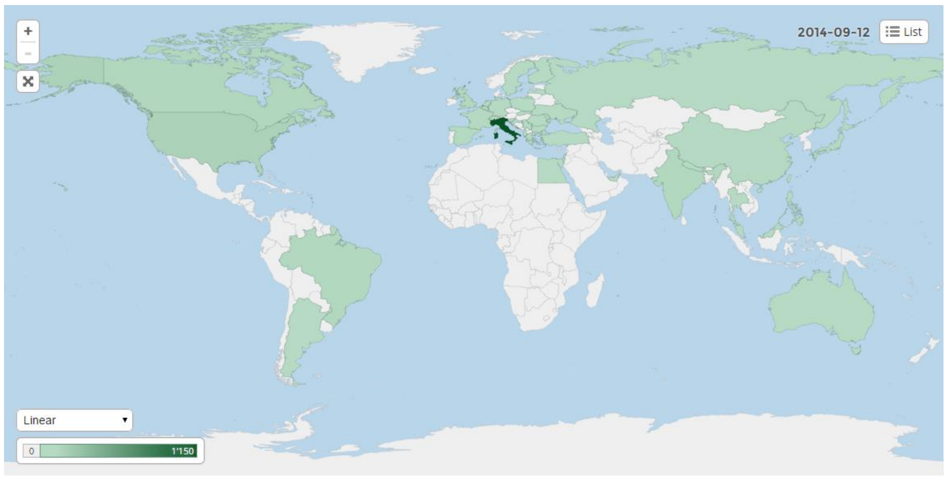


Figure 3.7: The linear color function of a map, which is displaying a dataset of one day.

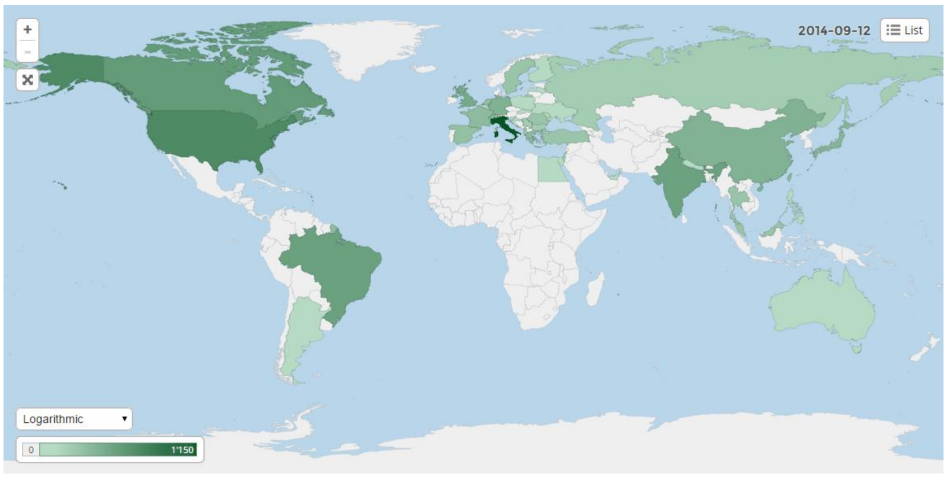


Figure 3.8: The logarithmic color function of a map, which is displaying a dataset of one day.

3.3.2 Map Navigation

In Figure 3.9 a country on the map is selected, whereupon the number of observed peers will be shown together with the maximal swarm size from peers of the selected country on the selected day. As a third value, the percentage of the observed peers to the sum of observed peers is displayed. It also shows the name of the country and the continent to which the country belongs.

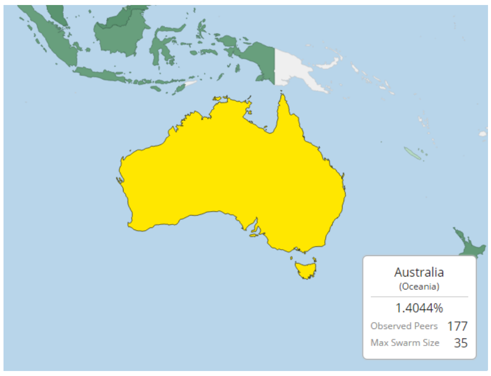
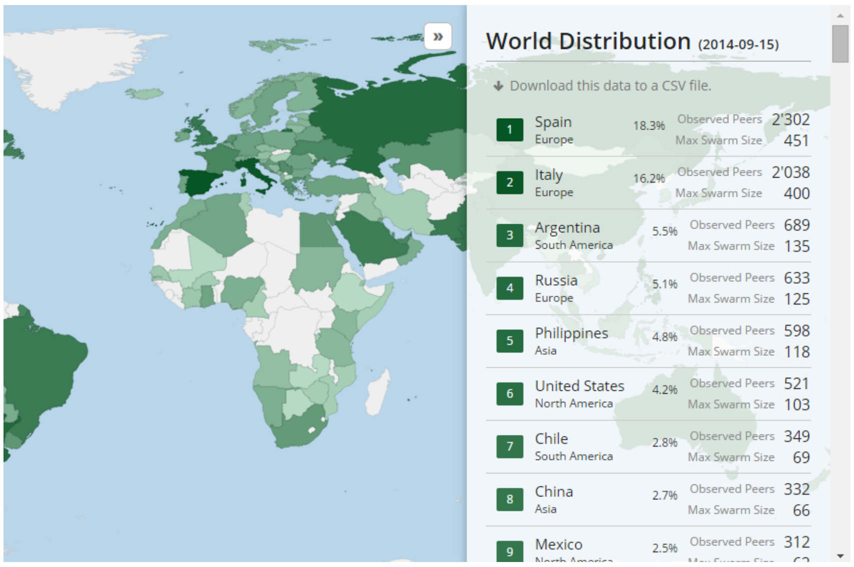


Figure 3.9: Australia as an example for a selected country.

When clicking on the "List" button, a collection of all countries slides over half of the map (Figure 3.10). The list presents all the data which is available for the selected day. It shows the number of observed peers, the maximum swarm size per country and the percentage value. These entries are equal to the ones which appear after selecting a country on the map. If a user has already selected a country on the map, it will be visually emphasized in the list.

There is also the possibility of downloading all the listed data in a CSV file for further examination.



World Distribution (2014-09-15)

Download this data to a CSV file.

1	Spain Europe	18.3%	Observed Peers Max Swarm Size	2'302 451
2	Italy Europe	16.2%	Observed Peers Max Swarm Size	2'038 400
3	Argentina South America	5.5%	Observed Peers Max Swarm Size	689 135
4	Russia Europe	5.1%	Observed Peers Max Swarm Size	633 125
5	Philippines Asia	4.8%	Observed Peers Max Swarm Size	598 118
6	United States North America	4.2%	Observed Peers Max Swarm Size	521 103
7	Chile South America	2.8%	Observed Peers Max Swarm Size	349 69
8	China Asia	2.7%	Observed Peers Max Swarm Size	332 66
9	Mexico	2.5%	Observed Peers	312

Figure 3.10: The list which slided over the map shows the aggregated data for all countries.

The map can also be displayed in fullscreen mode, where the size of the map increases to the full width and height of the window. It is recalculated and shown in an appropriate size. With a click on the red close button, it resizes back to the normal view.

The map adapts to the browser window on initialization. It reacts on resizing of the browser window, recalculates its size and redraws it completely. For further information see Chapter 3.6.

3.3.3 Map Dataset

The generic map shows the summary of information about all collected torrents from the last aggregated day, which is usually the day before. It displays an overview of the BitTorrent activity in a single country. The generic map is inserted into the project description page in order to give a first impression of the project.

The same map is shown in the detailed view with the only difference being the dataset, which only contains peers from one single torrent per day.

3.4 System Status

The system status or status page represents the current status of the Kraken Core software. It contains a table with information that is being directly delivered by the master. Precisely, the master delivers the number and ip address of every current slave, together with their status. The slave's status contains the number of messages sent and received, as well as the number of messages in the send/receive queue.

The system status is a helpful tool for the Kraken Core software administrator to see what the system is currently doing, all on one page. It is not necessary anymore to connect to the master via shell as you can see all the relevant information online. The pageable table opens the possibility to add multiple slaves without having a very long list on one page.

Since the information is only valuable to the administrator of the Kraken Core software, it is commonly expected for such information to be hidden in a secured environment. However, the data of the system status is not confidential. Therefore, a login would not be necessary, which is why the system status is publicly available. The entry in the website's menu is also less significant, as it is not useful for anyone apart from the system administrators.

3.5 Navigation

The navigation throughout the window is depending on the modules described in the beginning of this chapter. As a starting page, the project information and the general map were chosen. It gives the users some insight on the project. They are able to navigate to the other pages by making use of the menu on the top of the page (see Figure 3.11). The menu includes links to both the torrent list and the system status, whereas the specific torrent details page is only reachable via the selection of a torrent from the torrent list.

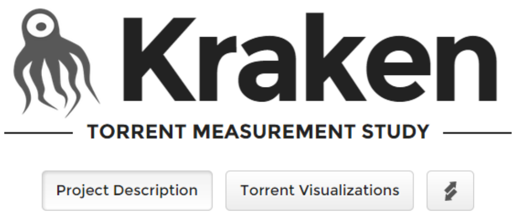


Figure 3.11: The navigation menu and title of the Kraken Web Interface.

3.6 Mobile Optimization

The Kraken Web Interface has been optimized for mobile devices. The appearance of the website depends on the width of the browser window or mobile screen port. If the browser window is larger than 1'200 Pixel, the website will be displayed in a centered bar with a width of 1'200 Pixel. If the window is smaller, the website adapts to the full screen width. The columns of the description texts as well as the table entries adapt to the smaller space by rearranging their content. The menu also rearranges to full width buttons if the window becomes even smaller.

This concept of a responsive website is commonly propagated to address mobile and tablet devices. Therefore, the Kraken Web Interface is fully mobile optimized. It can be accessed with a smartphone or a tablet as well as with a device with a larger screen. See Figure 3.12 for a view on the Kraken Web Interface project page on different devices.

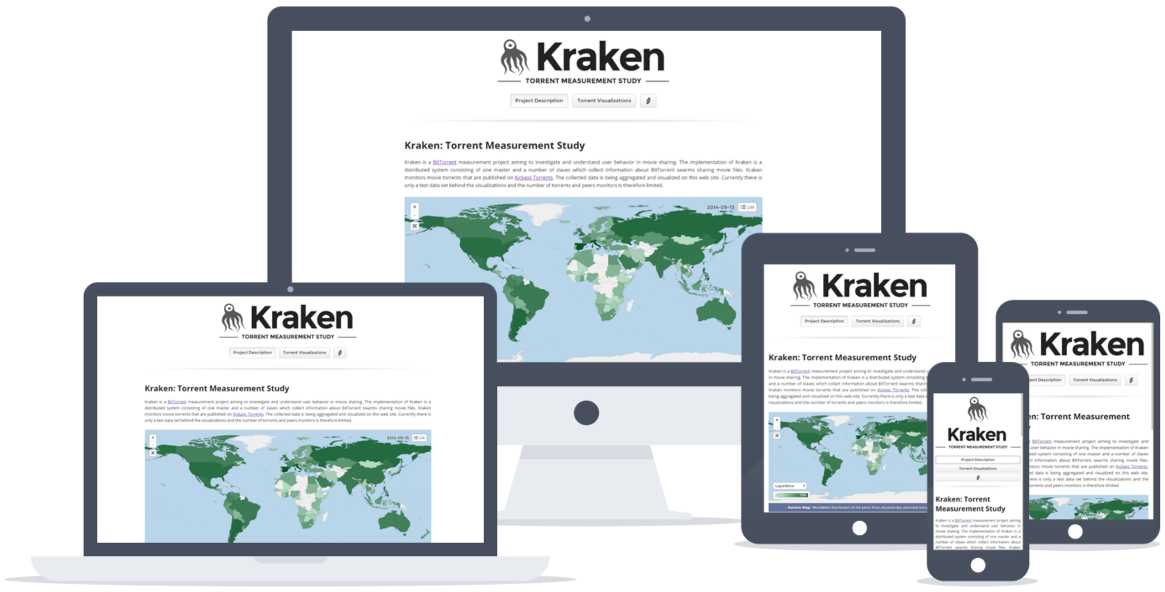


Figure 3.12: The Kraken Web Interface on different devices. Image Source: [47].

Chapter 4

Architecture

All data provided by the Kraken Web Interface is being collected by the Kraken Core software. The Kraken Core downloads the newest torrent files from the Kickass torrent platform. The tracker URI of each torrent is spread to all slaves, which then obtain a set of ip addresses. These addresses are transmitted to the master. The master collects the data and stores it into the database. The Web Interface database is connected to the Core Database with a federated engine. Every night, the new data is being aggregated and transmitted to the Web Interface database.

The status information is directly transferred from the master to the web server through a socket connection. It uses the same protocol as in the communication with the slaves.

Upon entering the website, the user receives data from both the database and the buffer of the socket connection, whilst being unaware of the source of the data. Figure 4.1 shows the architecture of the Kraken infrastructure.

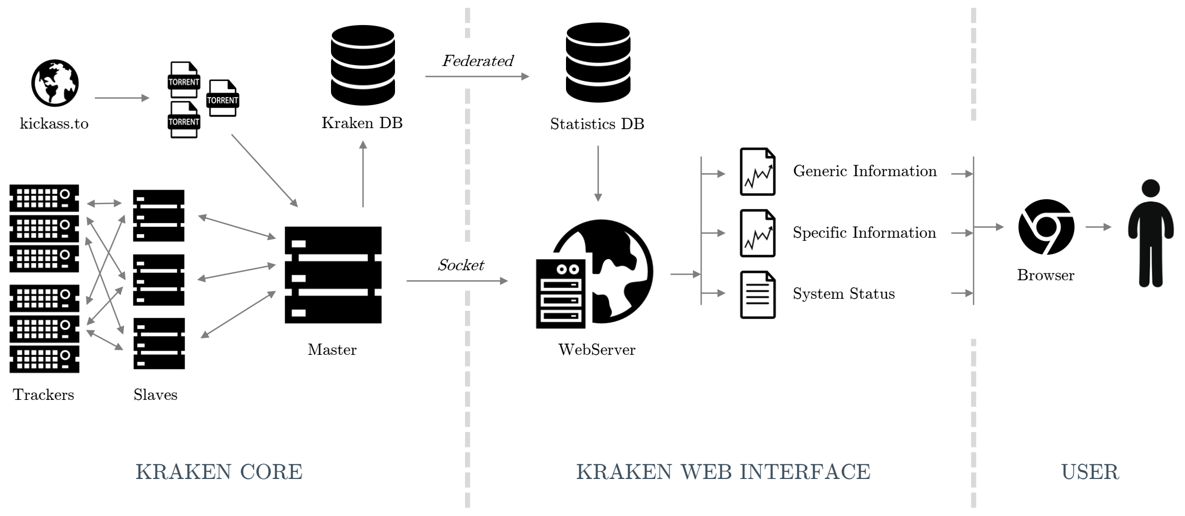


Figure 4.1: Architectural overview of the Kraken system. Image Icons: [27, 31, 52, 35, 24, 32, 26, 34, 42, 29]

4.1 Data Aggregation

In order to collect the greatest possible amount of unique peers, the Kraken Core software repeatedly connects to the tracker of the same torrents in short time intervals, and each time collects a random set of ip addresses. The outcome of this behavior leads to redundancy. To extract the relevant information out of the large database, the data needs to be aggregated.

4.1.1 Kraken Core Database

The Kraken Core database consists of three tables, where all gathered data is being stored. The table "TORRENTS" stores the metadata of every obtained torrent file, the table "ANNOUNCE_RESULT" stores every single request from the Kraken Core to a tracker of a torrent, and the table "PEERS" consists of every single collected and localized ip address.

For each torrent there are multiple requests to the tracker by multiple slaves, which then return a list of ip addresses. This leads to a sort of distribution of the data in which the table "PEERS" contains the most entries.

ANNOUNCE_RESULT	**PEERS**	**TORRENTS**
ID	ID	INFO_HASH
TRACKER_URI	IP_ADDRESS	ACTIVE
INTERVAL_NUMBER	PORT	TORRENT_TITLE
ANNOUNCE_COMPLETED	ASNUMBER	TORRENT_SIZE_KB
SEEDERS	CONTINENT	TORRENT_TRACKER_COUNT
LEECHERS	COUNTRY	TORRENT_COMMENT
TOTAL_PEERS	CITY	PUBLISH_DATE
RETURNED_PEERS	LATITUDE	MAGNET_URI
SLAVE_IP	LONGITUDE	TIME_ADDED
SLAVE_PORT		TIME_DEACTIVATED
LOCAL_TIMESTAMP		TORRENT_LINK
TIMESTAMP		

Table 4.1: Kraken Core database structure.

4.1.2 Reduction of Duplicated Data

Interesting for data visualization are mainly the activity per peer and the summed up activity per country. The peers table contains many duplicated entries due to aforementioned reasons. The activity per peer needs to be in a certain time range, which was determined to be one day long. From a perspective of an average torrent lifespan of around 30 to 300 hours [41], one day seemed to be a reasonable scale.

In the daily process of aggregation, the entries of the table "PEERS" were reduced to match the following criteria:

1. In a dataset of one day, one peer is only present once , except *2*.
2. It can be present multiple times, if it was observed in down- or uploading multiple torrents. The number of occurrences needs to match the number of torrents in which the peer has been participating.

The entries are reduced automatically every day at midnight and stored in a separated database. This scheduling task is being done with an event scheduler. The connection between the databases is established as a federated connection. See Appendix A for the SQL statement for the data aggregation.

The aggregated database also contains three tables including the ip addresses and some additional metadata. Table 4.2 shows an overview of the tables of the aggregated database.

The entries of all tables are being copied. This procedure produces a redundancy between the Kraken Core database and the aggregated database. It is legitimate though, because the entries of the core database will be archived when a torrent is not active anymore, whereas the aggregated database needs to have the data of archived entries present.

statistics_torrents	**statistics_torrentmeta**	**statistics_peers**
info_hash	id	id
title	observed_peers	ip_address
filesize	max_swarm_size	country_iso_code
publish_date	seeder_quota	info_hash
	info_hash	date
	date	

Table 4.2: Aggregated database tables for statistical and visualization purposes.

Chapter 5

Implementation

This chapter discusses the implementation of the design and the architecture of the Kraken Web Interface. In the section about the structuring of the JavaScript Code, there will be described that the JavaScript front-end modules are structured in a pattern using self-invoking functions which allows the usage of private variables and methods. There will follow a section about the build process including deliberations on how the CSS preprocessor Sass and its framework Compass were integrated into a Maven build process. In the next section, the map data generation will be illustrated, with remarks on how the raw data needed to be converted to a superset of the JSON format. In the following section on the Java back-end, the functionality of the back-end as an intermediate between the database and the front-end will be discussed. It also provides an insight to the connection between the Java back-end and the Kraken Core. To continue, the functionalities of the torrent list will be explained in detail and following that, all the functionalities of the map will be described. The chapter will conclude with a section on the time charts.

5.1 JavaScript Code Structuring

The JavaScript files were managed in the so-called JavaScript Module Pattern [17]. The benefit of using this pattern lies in the gain of modular structure in the otherwise unstructured JavaScript code.

It is possible to use private variables in JavaScript with the module pattern. A module consists of one function, assigned to a scope variable. This outer function defines some variables and inner functions in its own scope and only returns the ones which are to be accessed from the outside. The rest of it stays private.

The JavaScript Module Pattern works with self-invoking functions. After the definition of a function it will be directly executed. In the function parameters, libraries and other modules can be passed by as dependencies. These dependencies are renamed according to the function signature and can then be used inside the function in shorter variables.

One example of the JavaScript Module Pattern is the simplified JavaScript module of the system status page in Listing 5.1.

```javascript
kraken.systemstatus = ( function($, moment, d3, errorHandling) {

   /*
    * System Status
    * -------------
    * This module loads the data from the system
    * status servlet into the HTML table.
    *
    */
   "use strict";

   var servletUrl = null;

   function convertTimestamp(timestamp) {
      [...]
   }

   function init(servletUrlNew) {
      servletUrl = servletUrlNew;
          [...]
   }

   [...]

   return {
      init: init
   };

}(jQuery, moment, d3, kraken.errorHandling));
```

Listing 5.1: Simplified *systemstatus.js* as a sample for the Module Pattern.

The module in Listing 5.1 is depending on *jQuery*, *moment*, *d3* and the Kraken module *errorHandling*. Those modules can then be directly used inside the function. The required dependencies are clearly declared.

The module returns only one function called *init()*. This function acts as a constructor and is public, because it is returned by the module. It can be accessed from the outside by the call *kraken.systemstatus.init(url)*. The constructor assigns its parameter *servletUrlNew* to the instance variable *servletUrl*. However, *servletUrl* is a private variable, as it is not returned by the module. The method *convertTimestamp(timestamp)* is neither returned by the module and therefore private as well.

5.2 Build Process

The build process is managed by Apache Maven. The main task in the Kraken Web Interface besides the compilation of Java files is the build of the web resource files. The style files are written in the Sass syntax, relying on the Compass framework and the JavaScript files need to be concatenated. Both resulting output files need also to be minified for a better web page performance. Another part of the build process is the deployment to a standalone web server that is reachable in the internet.

5.2.1 Sass/Compass & Minifying

This section treats the build process of Sass and Compass. Firstly, an overview will be given of two possible solutions: jRuby together with the official Sass/Compass version and Wro4j. Eventually arguments in favor of and against the two solutions are discussed and the final decision is presented.

Sass and compass are both written in Ruby. To be able to compile the Sass code to regular CSS styles in a Maven process, it was necessary to either install Ruby on the build machine or produce an executable Java file, which will then run in the process. In order to be independent of installed software on a build system, a Java solution is the better option.

Wro4j [50] is a library, capable of these requirements. It supports Sass, is runnable in Maven and exists also as a runtime solution, whereas the files are compiled in runtime. On the downside, it neither supports Compass nor does it disallow the compilation of JavaScript files in a specified order. If an update of Sass is released the build also depends on the update cycle of Wro4j.

An alternative solution to Wro4j is using jRuby and the original Sass and Compass release. jRuby acts as a wrapped Ruby instance in a Java environment. The gems for Sass and Compass can be directly installed into the jRuby .jar file with just a few commands (see Appendix B for further instructions). The resulting .jar file is thus able to compile the Sass code with all Compass libraries to regular CSS code. It uses the official Sass and Compass distribution and can be easily updated. It is independent from installed software on a build system and easily integratable into a Maven build process. However on the downside, the compiling duration is rather high and the jar file is not managed by Maven. Wro4j also supports Minifying of CSS and JavaScript files, which is not supported within this solution.

Minifying JavaScript and CSS leads to a large decrease in the file size [49, 55]. It is therefore strongly recommended for a performant web application to minify the resources. Wro4j, the web resource optimizer for Java, has an integrated JavaScript and CSS minification unit. The compression is done with the YUI compressor, an open source JavaScript and CSS compressor. The YUI compressor can be also be used as a standalone JavaScript library. It is easily integratable into a Maven build process, but not able to concatenate JavaScript files.

Decision against Wro4j

The benefits of Wro4j, an integrated solution, did not overcome the drawbacks of its usage. Also, the solution of using the official compiler and compressor was convincing.

As the YUI compressor was not able to concatenate files, a file operation during the build process was necessary. The order of JavaScript files is important because of the dependencies between the JavaScript modules. For those reasons it was also necessary to have a list of file names deposited in a text file, which is then responsible for the concatenation order.

In Maven, basic file operations are only possible through additional plugins. In Apache Ant it is significantly easier to work with file based operations. An Ant build file can be easily executed by Maven with the Maven-Antrun-Plugin [3]. It is also possible to run Ant files automatically on a save operation in Eclipse. Hence, it is preferable to use an Apache Ant builder for concatenating the JavaScript files before the minification.

In order to combine all the web resource optimization in one place and to be able to use the resource optimization in an automatical build action in Eclipse, experience has shown that it is most reasonable to run all the resource optimization in one Ant script.

During the process of development, a minification of JavaScript and CSS is not useful. In a browser, debugging is much more complicate, if not impossible, with minified code. For that reason, the Ant script only minifies the code, if the ”productive” variable is set. This variable is set with the Maven profile ”productive”.

5.3 Map Data Generation

The visualization of the torrents on a map requires geographical map data of the entire world. Natural Earth Data, a public domain map dataset in various resolutions, created by many volunteer contributors of the North American Cartographic Information Society (NACIS) provides world-wide map data. The data is provided in a shape file together with political and statistical metadata.

For the generation of a map, D3.js expects map data in the JSON format, particularly GeoJSON or, with an additional JavaScript library, TopoJSON. The creator of D3.js provides a command-line tool, installable via NPM which is able to convert from all common formats to Topojson. This identically named software also supports shape files and GeoJSON as source.

In Figure 5.1 a progress overview of the map data generation is depicted. It starts by converting the shape file with the TopoJSON command-line tool to a TopoJSON file. The tool offers a lot of configurable options. The properties, that were already part of the shape file, could be partially removed, the resolution could be decreased etc. [10]. In the Kraken Web Interface, country name, continent and country ISO code remained unchanged during the generation. The generated TopoJSON file and the aggregated data from the database could then be used as an input for D3.js and the TopoJSON extension to visualize the map.

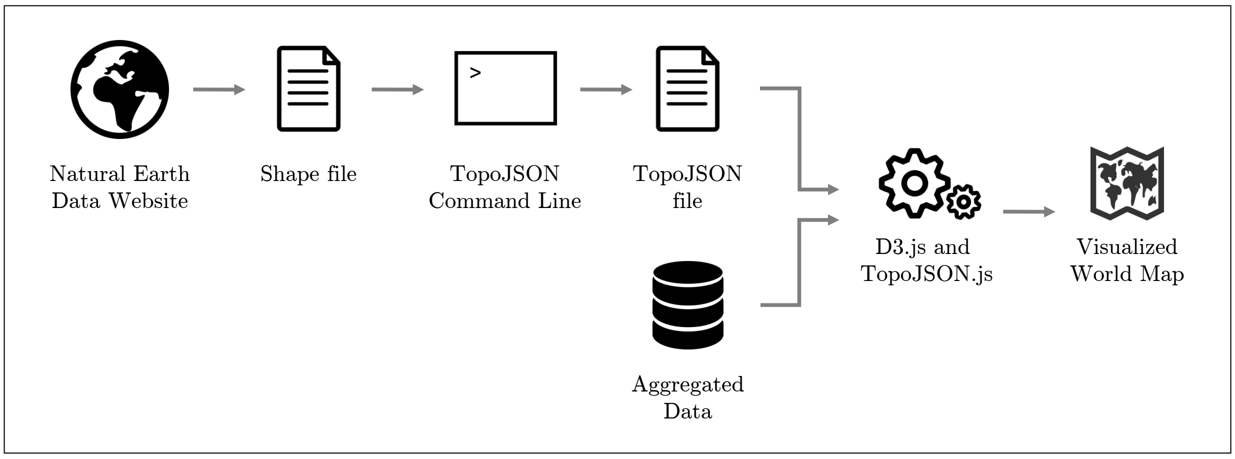


Figure 5.1: Conversion of the map data with TopoJSON. Image Icons: [43, 34, 60, 24, 28, 44]

5.4 Java Back-End

Since the Java back-end mainly acts as proxy between database and front-end, a main task of it is the mapping between the database and the JSON format. The Java back-end does not contain much logic and does not model every database entity. Therefore, it was reasonable not to use powerful frameworks as Gson [40] for the mapping. JSON Simple [45] however supports the simple, programmatic mapping between java objects and JSON. It was used to create JSON strings while looping through the returned results of the database.

The connection between the MySQL database and the Java back-end is established through the standard Java Database Connectivity Interface (JDBC). In order to keep the back-end simple, the SQL queries are directly stored in the database controller and similar to the JSON mapping, no powerful framework like Hibernate was used for calls to the database.

Following, the data requested from the database and mapped into the JSON format is then provided to the front-end with a Java Servlet. A servlet extends HttpServlet and provides a method which is called when the user requests the specified URL. This method then returns the queried data.

5.4.1 Socket Connection to Kraken Core

The information about the system status is not provided by the database but by the Kraken Core directly. A socket connection permanently runs between the web server and the master of the Kraken Core. The information is forwarded through a servlet directly to the front-end, since it is already received in the standard JSON format from the Kraken Core. Because the data is published publicly there is no need for strong security standards. Security could simply be provided by a firewall or other means.

In the front-end, the data is loaded with an Ajax call, that is standardized by jQuery. The data is then displayed in a standard HTML table filled in by *jQuery template*. It can be reloaded by clicking on the refresh icon.

5.5 Torrent List

The torrent list provides a filterable list of observed torrents, limited to one day. The list is filled with jQuery Template.

The day can be selected by clicking on the input field of the day on the top right of the table. A click on the field opens an inpage popup, where the user can select a day in a calendar view. The calendar has clickable days in the range of the first dataset in the database to the last. The calendar view is provided by jQuery pickadate. The day can also be changed by clicking on one of the two arrows on both sides of the day field. By clicking one of the arrows, the next or previous day is calculated with moment.js and passed to the API of jQuery pickadate.

The list can be filtered by keyword. When entering a keyword, it will be searched for in every word of every torrent. The keyword is split up on a space and it looks for every single keyword similar to SQL OR statement. The advantage of this technique is its accuracy: If an error is in one keyword, there are still results for the others. The disadvantage of this technique is however, because it does not capture all keywords at once, the user can be presented with a lot of search results. For example, if you enter "en", it leads to many results, because they are not filtered by the second or third keyword.

5.6 The Map

The core of the map module mainly consists of the integration of the D3 map functionality. D3 supports the simple creation of maps when providing map raw data in the format of GeoJSON or with an additional extension TopoJSON. A projection type is also required. It determines the way of how the map is modeled in two dimensions. See Figure 5.2 for a depiction of the twelve possible projection types in D3, which are part of the D3 core. In Kraken Web Interface, the equirectangular projection map was selected. It provides a well-known form of a map and is also the most simple projection using the identity function to calculate the map [8].

The plain map is merged with the data of the back-end. It was joined by matching the country ISO codes of the raw map data and the Kraken data. The additional data is attached to the D3 data model of the SVG map. Before rendering the map, the maximal value of the dataset of the observed peers is stored. The fraction of each of the country is calculated to produce a percentage value for the calculation of the color of each country. Eventually, the percentage value is transformed according to the selected color function (see Section 3.3.1) to an effective color of a country. The result is displayed in Figure 5.3.

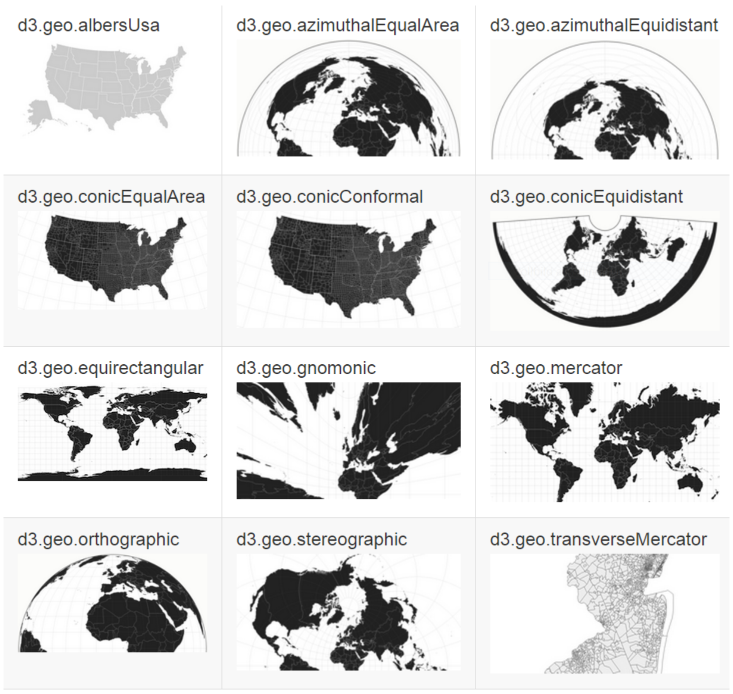


Figure 5.2: An extract of possible projection types in D3. Image: [8]

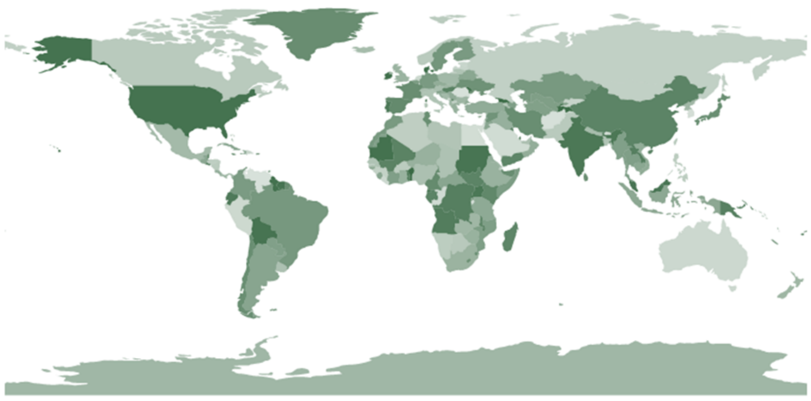


Figure 5.3: The colored map of Kraken Web Interface without any additional controls or functions.

In addition, the plain map offers a zooming feature. D3 already provides a default zooming, which reacts to mouse events. Usually, there are two buttons on a map which allow the zoom by clicking a button instead of mouse events. This is a handy feature for mobile devices because of the absence of a mouse. The zoom buttons also indicates that there is actually the possibility to zoom into the map.

The ”Click-to-Zoom” feature is not natively supported by the D3 library. With their open API, it can be added with reusing all the existing functionalities of D3. By adding this kind of feature, calculations of the position of the current viewport are inevitable. A viewport is a zoomed extract of the map, which does not show the full map anymore. Figure 5.4 displays an example of it. The x and y values, which are illustrated in the figure, need to be recalculated every time the map is zoomed. Considering the model of the viewport in Figure 5.4, the zooming is the increase or decrease of the size of the viewport. The user expects the center of the map to be still in the center of the map after the zooming. Therefore, the current center of the map needed to be taken into account as well. Eventually, after recalculating the new position of the viewport, it shows the same center as before with a new zoom depth.

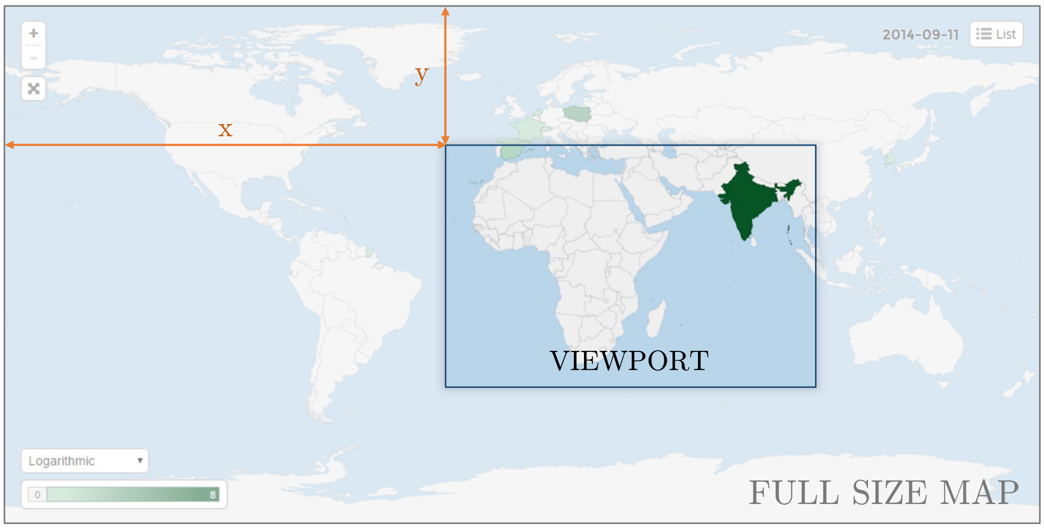


Figure 5.4: A zoomed viewport in relation to the full map. The x and y values represent the offset value of the viewport.

Zooming, dragging and dropping of the map leads to another problem, which is also not part of the native library of D3: The dragging of the map does not stop on any side of the map. This makes it possible to drag the map outside of the viewport. To prevent this from happening, a custom *move()* function was implemented. On every drag, it compares the current viewport in consideration of the current zoom depth to the maximal value, where the map is still inside the viewport.

The map is capable of being depicted in every size desired. It automatically reacts to changes of the browser width and redraws the map. Thanks to this functionality, it is also possible to open the map in fullscreen mode. The map redraws to the size of the browser viewport. One major difference to the regular view is however the possibility to drag the map outside of the browser viewport in fullscreen mode. When in regular mode, the map adapts its width proportionally to its height, while it behaves contrarily in fullscreen mode. In the fullscreen mode, the user expects a map which actually takes in the full height of the screen. If the map does not fit, it is not expected that either a scrollbar or a blank space on the bottom of the page appears, but that the map is draggable to the outside of the browser viewport. This scenario is illustrated in Figure 5.5.

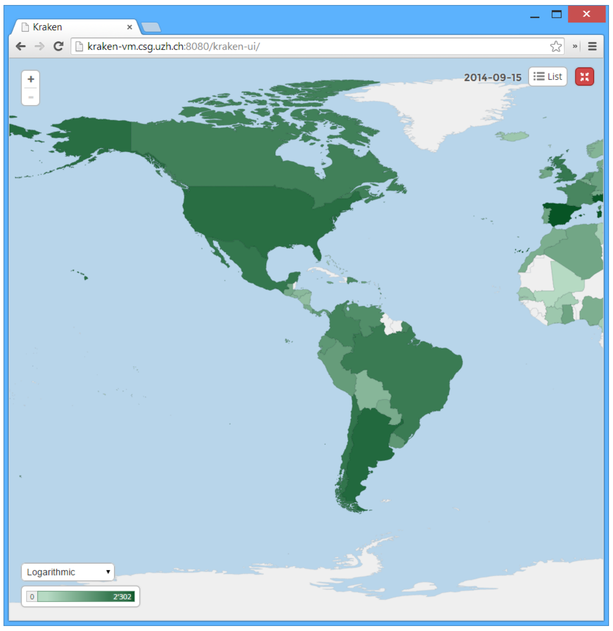


Figure 5.5: The map in fullscreen mode needs to fill the full height of the browser window.

The JavaScript module *map.js* offers an API which supports three different ways to initialize. Along with the regular initialization it is possible to initialize the module hidden or without controls. The hidden initialization prepares the map in a hidden version and assures that no side effects occur. The hidden controls are useful for very small map depictions where the controls overlay the map too much. The API offers also a way to disable the redrawing of the map on browser window width changes.

Furthermore, the map reacts to the two events ”mapInfoHashChange” and ”mapDateChange”, which both need to provide either the date or the info hash of a torrent. If one of these two events are fired to the <*body*>, the map reacts accordingly and adapts itself to the new parameters. An event based communication between the modules allows the strict separation of the modules. Those two events are mainly used for the communication between the time chart and the specific map on the Torrent Details Page.

5.7 Time Chart

The time chart is a configured instance of the C3.js chart library. C3.js relies on D3.js, but, in contrary to D3.js, it is not a programming library, but a configurable one.

The time chart of the Kraken Web Interface is a spline area chart. It depicts the peers and the fraction of seeders on each observed day. Each day is selectable. If selected, it calls a custom function, which loads the specific map of one movie torrent on the selected day and highlights the date. In order to optimize the visual impression, a spline area chart was selected. Another possibility for rendering the chart would be a line or line area chart with sharp edges. As it was pointed out in the comments of the evaluation questionnaire (see Appendix C.4), a spline chart may have some disadvantages in displaying the peaks on the exact date.

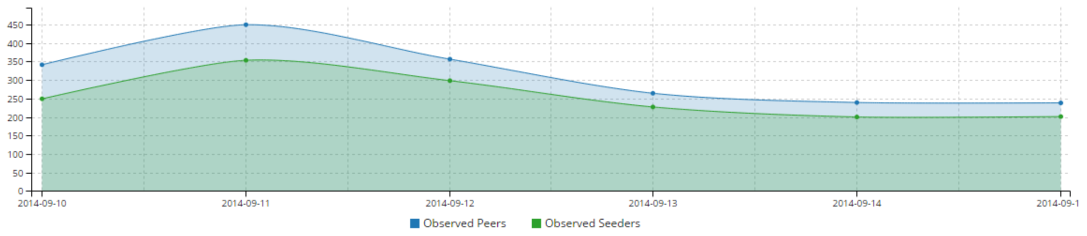


Figure 5.6: The time chart graph with a sample dataset.

The zoom feature of the chart is enabled, meaning that the user is able to zoom and drag and drop the chart. While adding the zooming feature, an issue evolved: The time chart jumped an inch to the right, when dragging the graph. This issue was reported as an official bug report [57] on the *GitHub Issues* website of the framework. The author of the framework reacted, and after a few written exchanges, the bug was fixed. Although a new bug had emerged from the primary bugfix, a new bug report [58] was resolved quickly. Eventually, there was a new version of the C3.js framework released, including both bugfixes.

Chapter 6

Usability Evaluation

Today, an evaluation of the usability of the software is very important. The usability is a leading factor for the user to decide whether to use the software on a regular basis. If they decide not to revisit a website because they did not find the desired information, the website has failed as a medium of information transmission. In this chapter, the usability of the Kraken Web Interface will be evaluated. It will begin with the scenario and setup of a questionnaire to gather data of the users opinion of the software. For this thesis, a standardized SUS questionnaire was chosen for determining a usability score. In the second section of this chapter, the responses to the questionnaire will be evaluated, before being analyzed and interpreted in the last section.

6.1 Scenario & Setup

The usability of the Kraken Web Interface was evaluated by a standardized System Usability Scale (SUS) questionnaire. The SUS questionnaire consists of 10 questions with a scale from 1 to 5. In this evaluation, the scale was stretched to a scale from 1 to 10, to get even exacter results. Table C.1 in Appendix C.1 lists the asked standard SUS questions.

The SUS questionnaire expects the user to be familiar with the software. For that reason, in the beginning of the questionnaire, there were six additional non-required questions to motivate the user to get familiar with the software. These questions were contentual and asked for detail facts, which could be found on different parts of the Kraken Web Interface. A response to these questions was optional. Table 6.1 lists the questions.

For statistical reasons, at the end of the survey there were five demographic questions and one input field for general comments. Two of those questions regarded the IT skills of the user and were required to answer, the other three questions were optional. In Table 6.2 they are listed.

The questionnaire was online conducted with Google Forms [39]. An e-mail with an invitation to participate at the questionnaire was sent to 28 persons. Additionally, the link to the questionnaire was released in two different Facebook groups with primarily members of the University of Zurich.

1.	Find the generic map with all the observed peers from one day in it. Find the torrent list, which is a list of torrents observed on one specific day. Click on one of them and find the torrent details page with a timegraph and a specific map.
2.	How many people in percentage were downloading from Italy on 2014-09-15?
3	On what day were the most people downloading the movie ”Subha Hone Na De Full Song 1080p HD”?
4.	Select the following movie and see what the distribution looks like on 2014-09-12. 82.4% of all downloads of this movie were conducted by people based in Italy. Change the color visualization on the bottom left to see how the color of the countries change by changing the function from Logarithmic to Linear. ”Chef La Ricetta Perfetta 2014 iTALiAN MD WEBRip XviD-FREE[MT] avi”
5.	Same Movie as in 4. Open the map list of the movie and download the CSV file containing all the data. Open the CSV file in a text editor or Excel.
6.	How many people from Switzerland downloaded the movie ”Al Filo Del Mañana [BluRay Screener][Español Castellano]” on 2014-09-11?

Table 6.1: Context questions before starting the questionnaire.

1.	Male/Female
2.	Name
3.	Age
4.	What is the highest level of education you have completed?
5.	Would you like to leave a comment?

Table 6.2: Demographic questions at the end of the questionnaire.

6.2 Results

In total, 19 persons completed the questionnaire, whereas 32% were women. Most of the probands were highly educated with a rate of 82% of the participants having at least a Bachelor degree. 15 out of 19 are working or studying in the IT sector, whereas the mean estimated IT skills of the users is 7.74 on a scale between 1 and 10. In the Appendix C.2 you can find in Table C.2 the responses to the SUS questionnaire. In Figure 6.1 you can find the statistical data about the usability evaluation visualized.

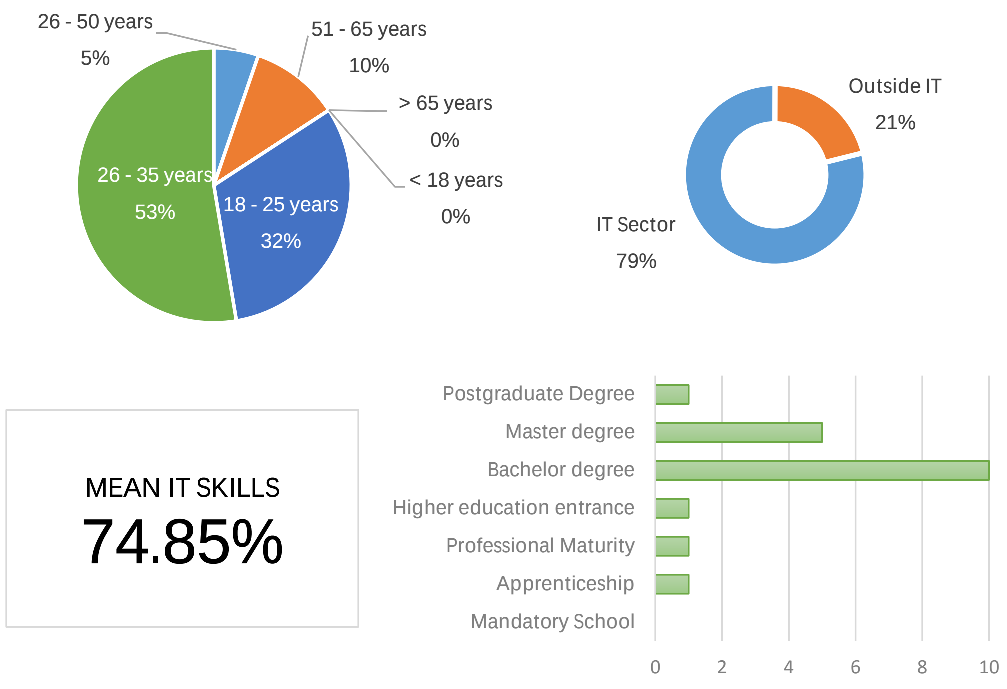


Figure 6.1: Visualized statistical data about the evaluation study participants.

The responses of the questionnaire were evaluated and rescaled to the original SUS scale between 1 and 5. Then, the SUS scores were calculated according to the rules of Brooke [12]. The calculation approach is in Appendix C.3.

The mean SUS score of the Kraken Web Interface attained 70.94 with a median value of 70. The bandwidth of the scores was between 33.33 and 94.44. In ten-point ranges, the most SUS scores were between 80 and 90. In Figure 6.2 you can see the full distribution of the SUS scores.

Women rated the software higher, than men. The mean SUS score of female participants was 74.07, whereas the mean SUS score of men was 69.49. The age of the participant is relevant for the SUS score, the youngest participants were most convinced about the software. Participants in the age of 18 to 25 rated with an average score of 82.41, whereas the older participant were all below the total mean SUS score. See Table 6.3 for the distribution among the age of the participants and number of participants in that group.

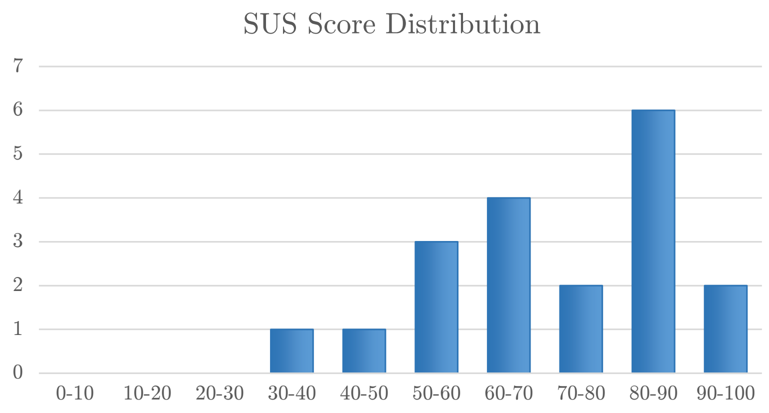


Figure 6.2: SUS score distribution in ranges of 10 out of 100 with the number of participants with a SUS score in that range.

Age	mean SUS Score	Nr. of participants
18 - 25 years old	82.41	6
26 - 35 years old	65.33	10
36 - 50 years old	67.78	1
51 - 65 years old	66.11	2

Table 6.3: Distribution of the SUS scores among the age of the participants.

The SUS score of the 15 persons who are working or studying in the IT sector is minimally lower than the total average: 69.41. However, the score of the people who are not related to the IT sector is with 76.67 higher than the total average.

90.35% of the optional context questions were answered correctly, whereas just 1.75% of the questions remained unanswered. Every participant responded to the optional questions. The mean SUS score of the 8 participants with at least one faulty answer in the context questions is with 68.89 not significant smaller than the total average SUS score.

6.3 Interpretation and Analysis

According to the adjective-mapping of the SUS score, a mean score of 70.94 is closest to the word ”good” [6]. According to the same study, it is slightly above the average value of SUS scores evaluating websites, which is 68.2.

Most attentive in this evaluation is the large difference of SUS scores between the young participants (18-25 years old) and the older ones ($>$ 25 years). One possible explanation might be that the younger the participating persons are, the more natively they handle digital media. Digital natives can be described as a group of people who are born after 1980 and grew up with digital technologies [51]. The second group of persons (25-34 years old) would also match the criteria of being digital natives. However, this group shows the largest anomalies in the data, as in this group the two lowest SUS scores were measured.

The people, who are not studying or working in the IT sector rated the software higher than the ones, related to the IT sector. A possible explanation for this phenomenon might be their lower knowledge about the matter of BitTorrent. If the people do not know much about the context of the software, they might allege their unclarity to their lack of knowledge and not to possible unusable parts of the software. Another reason might be, that people, which are not related to the IT do matter about the context of the software that much, but rather focus on the visual part of it, what might influence the resulting SUS score.

The context questions in the beginning may also have had an impact on the SUS scores. A question like the sixth, where the participant was encouraged to answer how many downloads were made on a day, may confuse people who are not familiar with BitTorrent. On the website the term ”download” was only used in the project description, but not on the map or map list. Only the term ”Observed Peers” was written next to the looked for number. The evaluated data however showed a different picture. Only one of six participants, who did not answer or answered the sixth question incorrectly, was effectively a person without a relation to the IT sector. This these could not be ascertained.

The anomalies in the data can be explained by the comments the respective participants left behind. Their biggest concern was the functionality of the search bar, which did not deliver the expected results. Another concern was the navigation of the page, which was not clear to everybody. These matters may have impacted the ratings in the SUS score negatively.

Scores for individual items are not meaningful on their own, according to Brooke [12]. This disallows a further examination of the single questions and how a SUS score is achieved.

Chapter 7

Summary, Conclusions & Future Work

This chapter concludes the thesis of the Kraken Web Interface. There will be a summary of the results of the work on the Kraken Web Interface, followed by conclusions on how this thesis might have an impact on future torrent research. Eventually, there will be some thoughts of how the Kraken Web Interface could be further developed.

7.1 Summary and Conclusions

As the main result, a web interface which visualizes the aggregate Kraken Core data has evolved from this Bachelor Thesis. It continuously interprets the current data and visualizes the aggregated data as information in three disparate modules: ”Specific information”, ”Generic information” and ”System Status”. There are mainly two visualizations present on it: A map with either generic or specific information and a time chart, which visualizes the activity of one single torrent over time. The resulting Kraken Web Interface is currently deployed on a server of the IFI at the University of Zurich with a mock dataset of one week:

http://kraken-vm.csg.uzh.ch:8080/kraken-ui/

The data aggregation has transformed the raw data, which has been randomly collected by the Kraken Core application, into a meaningful form. It is now being used in the specific and generic map, but also in the time chart.

The general map depicts worldwide BitTorrent traffic. The distribution on the map indicates how many peers have connected to the network in general and from which locations. The user can identify the country in which the peers are most active in down- or uploading, and the countries, in which the peers are not active at all.

The specific maps of the torrents allow for researchers to observe the changes in the geographic locality of one specific torrent file in the course of multiple days. The according time graph delivers the data on the mass of downloads over time. Researchers can now understand the development of BitTorrent traffic both in relation to geographic locality and in absolute figures.

The time chart depicts the progression of the quantity of peers of one single torrent on one day. The user can select a day, and the map will refresh with the selected dataset.

The Kraken Web interface can have an impact on the BitTorrent research community, because the gathered data of the Kraken measurement study has now been made publicly available. Researchers from all over the world are now able to look up the data of the Kraken Web Interface to further study the activities in a BitTorrent network.

7.2 Future Work

Currently, the torrent list and the specific map are two separate pages. User comments, which were gathered in the evaluation questionnaire, have shown that the placement of the torrent list and the torrent details page on separate web pages causes confusion to some users. All user comments are listed in Appendix C.4. A reasonable variation of the specific information section could include a merging of those two pages. The torrent list could be exchanged by an autocomplete input field, which would open the map and the time chart directly on this page. The selection of the torrent and the depiction on the map would remain on one page. This change would also simplify the main navigation, as there would only be one page behind the button "Torrent Visualization", instead of the torrent list and the torrent details page.

The map currently supports only a depiction of data of one single day. The map needs to be redrawn upon switching to another day, e.g. after selecting another date in the time chart. A possible improvement of the Kraken Web Interface could involve an adaption of the map to a multi-day view. The map could involve a range selector within the lifespan of the depicted torrent. This extension would increase the interactivity and user experience of the map.

In further research, the localized and aggregated data of the Kraken Web Interface might also be used to improve the locality principle. In this context, locality means that it is preferable to share files in the same or close network, rather than in distant networks. Keeping traffic local is generally less expensive for the Internet Service Providers (ISP). The aggregated data could be downloaded via CSV or directly withdrawn from the Kraken Web Interface database in order to further examine the locality.

Bibliography

[1] Amsul. jquery pickadate. `http://amsul.ca/pickadate.js`. [Online, accessed 2014-11-30].

[2] Apache. The ant project. `http://ant.apache.org`. [Online, accessed 2014-11-10].

[3] Apache. Maven antrun plugin. `http://maven.apache.org/plugins/maven-antrun-plugin`. [Online, accessed 2014-11-29].

[4] Apache. Official maven website. `http://maven.apache.org`. [Online, accessed 2014-11-10].

[5] Stefan Matthias Aust, Anders Bengtsson, Geert Bevin, et al. jruby: The ruby programming language on the jvm. `http://jruby.org`. [Online, accessed 2014-11-10].

[6] Aaron Bangor, Philip Kortum, and James Miller. Determining what individual sus scores mean: Adding an adjective rating scale. *Journal of usability studies*, 4(3):114–123, 2009.

[7] Aaron Bangor, Philip T Kortum, and James T Miller. An empirical evaluation of the system usability scale. *Intl. Journal of Human–Computer Interaction*, 24(6):574–594, 2008.

[8] Mike Bostock. Geo projections in d3.js. `https://github.com/mbostock/d3/wiki/Geo-Projections`. [Online, accessed 2014-12-03].

[9] Mike Bostock. Topojson. `https://github.com/mbostock/topojson/wiki`. [Online, accessed 2014-11-10].

[10] Mike Bostock. Topojson: Command-line tool. `https://github.com/mbostock/topojson/wiki/Command-Line-Reference`. [Online, accessed 2014-11-17].

[11] Mike Bostock. Topojson project description. `https://github.com/mbostock/topojson/wiki`. [Online, accessed 2014-11-15].

[12] John Brooke. *SUS-A quick and dirty usability scale*, volume 189. London: Taylor & Francis, 1996.

[13] Paul Burgess. jquery template. `https://github.com/codepb/jquery-template`. [Online, accessed 2014-11-30].

[14] Howard Butler, Martin Daly, Allan Doyle, Sean Gillies, Tim Schaub, and Christopher Schmidt. Geojson format specification. `http://geojson.org/geojson-spec.html`. [Online, accessed 2014-11-10].

[15] Hampton Catlin, Natalie Weizenbaum, Christopher Eppstein, and numerous contributors. Sass: The official website guide. `http://sass-lang.com/guide`. [Online, accessed 2014-11-10].

[16] Xi Chen. Sass/compass installation guide with jruby. `http://seanchenxi.com/java/sass-compass-jruby-single-jar`. [Online, accessed 2014-12-01].

[17] Ben Cherry. Javascript module pattern: In-depth. `http://www.adequatelygood.com/JavaScript-Module-Pattern-In-Depth.html`. [Online, accessed 2014-12-02].

[18] Bram Cohen. Incentives build robustness in bittorrent. In *Workshop on Economics of Peer-to-Peer systems*, volume 6, pages 68–72, 2003.

[19] Bram Cohen. Bittorrent protocol specification. `http://www.bittorrent.org/beps/bep_0003.html`, 2008. [Online, accessed 2014-11-10].

[20] Designmodo. Icon: News notice. `http://www.flaticon.com/free-icon/news-notice_4442`. [Online, accessed 2014-12-02].

[21] Christopher Eppstein. Compass: An open-source css authoring framework. `http://compass-style.org`. [Online, accessed 2014-11-10].

[22] Kraig Finstad. The system usability scale and non-native english speakers. *Journal of usability studies*, 1(4):185–188, 2006.

[23] Carlos Flavián, Miguel Guinalíu, and Raquel Gurrea. The role played by perceived usability, satisfaction and consumer trust on website loyalty. *Information & Management*, 43(1):1–14, 2006.

[24] Freepik. Icon: Database. `http://www.flaticon.com/free-icon/database_1059`. [Online, accessed 2014-12-04].

[25] Freepik. Icon: Directory submission. `http://www.flaticon.com/free-icon/directory-submission-symbol_48671`. [Online, accessed 2014-12-04].

[26] Freepik. Icon: Document with line chart. `http://www.flaticon.com/free-icon/document-with-line-chart_33279`. [Online, accessed 2014-12-04].

[27] Freepik. Icon: Earth. `http://www.flaticon.com/free-icon/earth_32445`. [Online, accessed 2014-12-04].

[28] Freepik. Icon: Engineering. `http://www.flaticon.com/free-icon/engineering_1850`. [Online, accessed 2014-12-04].

[29] Freepik. Icon: Frontal standing man silhouette. `http://www.flaticon.com/free-icon/frontal-standing-man-silhouette_10522`. [Online, accessed 2014-12-04].

[30] Freepik. Icon: News notice. `http://www.flaticon.com/free-icon/educational-graphic_42927`. [Online, accessed 2014-12-04].

[31] Freepik. Icon: Rack server. `http://www.flaticon.com/free-icon/rack-servers_31726`. [Online, accessed 2014-12-04].

[32] Freepik. Icon: Server with the earth. `http://www.flaticon.com/free-icon/server-with-the-earth_31553`. [Online, accessed 2014-12-04].

[33] Freepik. Icon: Spreadsheet cell. `http://www.flaticon.com/free-icon/spreadsheet-cell_31023`. [Online, accessed 2014-12-04].

[34] Freepik. Icon: Text document. `http://www.flaticon.com/free-icon/text-document_32329`. [Online, accessed 2014-12-04].

[35] Freepik. Icon: Torrent symbol file format. `http://www.flaticon.com/free-icon/torrent-symbol-file-format_28969`. [Online, accessed 2014-12-04].

[36] Freepik. Icon: World map. `http://www.flaticon.com/free-icon/world-map_62443`. [Online, accessed 2014-12-04].

[37] Nicolas Gallagher and Jonathan Neal. Normalize.css: A modern, html5-ready alternative to css resets. `http://necolas.github.io/normalize.css/`. [Online, accessed 2014-11-10].

[38] Rob Garrison and Christian Bach. jquery tablesorter. `http://mottie.github.io/tablesorter/docs`. [Online, accessed 2014-11-30].

[39] Google. Forms. `http://www.google.com/forms/about`. [Online, accessed 2014-12-06].

[40] Google. Gson library. `https://code.google.com/p/google-gson`. [Online, accessed 2014-11-29].

[41] Lei Guo, Songqing Chen, Zhen Xiao, Enhua Tan, Xiaoning Ding, and Xiaodong Zhang. Measurements, analysis, and modeling of bittorrent-like systems. In *Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement*, pages 4–4. USENIX Association, 2005.

[42] Icomoon. Icon: Chrome logo. `http://www.flaticon.com/free-icon/chrome-logo_23689`. [Online, accessed 2014-12-04].

[43] Icomoon. Icon: Earth. `http://www.flaticon.com/free-icon/earth_24390`. [Online, accessed 2014-12-04].

[44] Icons8. Icon: World map trifold. `http://www.flaticon.com/free-icon/world-map-trifold_24485`. [Online, accessed 2014-12-04].

[45] Json simple. `https://code.google.com/p/json-simple`. [Online, accessed 2014-11-29].

[46] Kickass torrents. `https://kickass.so`. [Online, accessed 2014-11-29].

[47] Ben Kroll. What is responsive web design. http://blog.dudamobile.com/what-is-responsive-web-design. [Online, accessed 2014-11-29].

[48] Yukihiro Matsumoto. Ruby, the programming language. https://www.ruby-lang.org/en/. [Online, accessed 2014-11-10].

[49] Alex Nicolaou. Best practices on the move: building web apps for mobile devices. *Queue*, 11(6):30, 2013.

[50] Alex Objelean and Bogdan Csoregi. Wro4j: Web resource optimizer for java. https://code.google.com/p/wro4j/. [Online, accessed 2014-12-05].

[51] John Palfrey and Urs Gasser. Opening universities in a digital era. *New England Journal of Higher Education*, 23(1):22–24, 2008.

[52] Picol. Icon: Server. http://www.flaticon.com/free-icon/server_14725. [Online, accessed 2014-12-04].

[53] Dongyu Qiu and Rayadurgam Srikant. Modeling and performance analysis of bittorrent-like peer-to-peer networks. In *ACM SIGCOMM Computer Communication Review*, volume 34, pages 367–378. ACM, 2004.

[54] Sandvine. Global internet phenomena report 1h 2014. https://www.sandvine.com/downloads/general/global-internet-phenomena/2014/1h-2014-global-internet-phenomena-report.pdf. [Online, accessed 2014-12-07].

[55] Steve Souders. High-performance web sites. *Communications of the ACM*, 51(12):36–41, 2008.

[56] Masayuki Tanaka. C3.js: D3-based reusable chart library. http://www.c3js.org. [Online, accessed 2014-11-10].

[57] Masayuki Tanaka and Sebastian Schrepfer. Bug report (a) of the c3.js chart library. https://github.com/masayuki0812/c3/issues/598. [Online, accessed 2014-12-03].

[58] Masayuki Tanaka and Sebastian Schrepfer. Bug report (b) of the c3.js chart library. https://github.com/masayuki0812/c3/issues/721. [Online, accessed 2014-12-03].

[59] David M. Theobald. Understanding topology and shapefiles. http://www.esri.com/news/arcuser/0401/topo.html, 2011. [Online, accessed 2014-11-10].

[60] TutsPlus. Icon: Line command. http://www.flaticon.com/free-icon/line-command_23399. [Online, accessed 2014-12-04].

[61] W3C. Official css website. http://www.w3.org/Style/CSS. [Online, accessed 2014-11-29].

[62] Tim Wood and Iskren Chernev. Moment.js: Parse, validate, manipulate, and display dates in javascript. http://momentjs.com. [Online, accessed 2014-11-10].

[63] Yahoo. Yui: Compressing javascript and css files. `http://yui.github.io/yuicompressor/`. [Online, accessed 2014-11-10].

Abbreviations

API	Application Programming Interface
BT	BitTorrent
DOM	Document Object Model
HTML	Hypertext Markup Language
IP	Internet Protocol
ISP	Internet Service Provider
JDBC	Java Database Connectivity Interface
JSON	JavaScript Object Notation
NPM	Node.js Package Manager
SQL	Structured Query Language
TCP	Transmission Control Protocol
URI	Uniform Resource Identifier
W3C	World Wide Web Consortium

Glossary

Back-End The back-end is a broad term. In this thesis, it is used for the collection of the data from the Kraken Core, its aggregation, rehashing and delivering to the front-end.

BitTorrent BitTorrent is a file sharing protocol with an own distribution network for every single shared file.

ClickListener An event listener on a DOM element. Reacts on clicking an element and runs a defined function.

Compass A framework for the Sass language, which e.g. simplifies the handling of browser vendor-prefixes.

Front-End The front-end is a broad term. In this thesis, it is used for the visual part of the software, thus the web site.

GeoJSON A format for encoding a various geographic data structures like maps.

GIS Software Geographic Information Systems Software; models geographic information.

GitHub A collaborative, distributed revision control software with source code management functionality.

Java An object-oriented cross-platform programming language which was used in the Kraken Web Interface to program the back-end.

JavaScript A script language, which is running in the browser and offering dynamic changes to a static HTML website.

jQuery A front-end web library, which simplifies the DOM manipulation by providing cross-browser functions.

jRuby An implementation of a Ruby interpretor in Java. In Kraken Web Interface it is used for running Sass.

Kraken Kraken is a torrent measurement study and software for analyzing the routes of the BitTorrent network.

Kraken Core Kraken Core is the software part of the Kraken measurement study. It involves the data gathering, but not the visualization of the data.

Leecher A peer in the BitTorrent network, which is mainly download data from the network.

Node.js A framework for building JavaScript back-end applications.

Ruby An object-oriented programming language which supports dynamic typing. In the Kraken Web Interface is used for the CSS precompiler Sass.

Ruby Gem Ruby Gem or simple Gem; a package in the Ruby programming language.

Sass A scripting language which is interpreted into CSS styles.

Seeder A peer in the BitTorrent network, which is mainly uploading data into the network.

TopoJSON An extension of the GeoJSON encoding standard, that supports storing shared borders only once.

Torrent A file sharing concept with the most comment implementation of BitTorrent (see BitTorrent)

Viewport The part of the window, which actually available for displaying content. In the context of a cartography, it is the section of the map which is currently visible (e.g. when zoomed in).

List of Figures

List of Tables

Appendix A

SQL Script

The following SQL script aggregates the data from the Kraken database and inserts them into into the statistics database of the Kraken Web Interface. It creates an event, which automatically triggers this aggregation shortly after midnight.

```
DELIMITER $$
CREATE EVENT `CREATE_DAILY_STATISTICS`
ON SCHEDULE EVERY 1 DAY STARTS '2014-09-30 00:03:00'
DO BEGIN

    -- yesterday's date
    SET @QUERYDATE = DATE(DATE_SUB(NOW(), INTERVAL 1 DAY));

    INSERT INTO `kraken-statistics`.statistics_peers
        (ip_address, country_iso_code, info_hash, date)

        (SELECT PEERS.IP_ADDRESS AS ip_address,
            PEERS.COUNTRY AS country_iso_code,
            ANNOUNCE_RESULT.INFO_HASH AS info_hash,
            @QUERYDATE AS date

        FROM `kraken-master`.ANNOUNCE_RESULT,
            `kraken-master`.PEERS
        WHERE ANNOUNCE_RESULT.ID = PEERS.ID
        AND DATE(ANNOUNCE_RESULT.TIMESTAMP) = @QUERYDATE
        GROUP BY PEERS.IP_ADDRESS,
            ANNOUNCE_RESULT.INFO_HASH);

    INSERT INTO `kraken-statistics`.statistics_torrentmeta
        (observed_peers, max_swarm_size, seeder_quota, info_hash, date)

        (SELECT observed_peers,
            max_swarm_size,
            seeder_quota,
            table1.info_hash,
            date

        FROM (
            SELECT MAX(TOTAL_PEERS) AS max_swarm_size,
                AVG(SEEDERS/TOTAL_PEERS) AS seeder_quota,
                INFO_HASH AS info_hash,
                @QUERYDATE AS date

            FROM `kraken-master`.`ANNOUNCE_RESULT`
            WHERE ANNOUNCE_COMPLETED = 1
            AND DATE(TIMESTAMP) = @QUERYDATE
            GROUP BY INFO_HASH
        ) AS table1, (
            SELECT COUNT(*) AS observed_peers,
                info_hash AS info_hash
            FROM `kraken-statistics`.statistics_peers
            WHERE date = @QUERYDATE
            GROUP BY info_hash
        ) AS table2

        WHERE table1.info_hash = table2.info_hash);

    INSERT INTO `kraken-statistics`.statistics_torrents
        (info_hash, title, filesize, publish_date)

        (SELECT INFO_HASH AS info_hash,
            TORRENT_TITLE AS title,
            TORRENT_SIZE_KB AS filesize,
            DATE(PUBLISH_DATE) AS publish_date

        FROM `kraken-master`.TORRENTS
        WHERE INFO_HASH NOT IN (
            SELECT info_hash
            FROM `kraken-statistics`.statistics_torrents)
        AND INFO_HASH IN (
            SELECT info_hash
            FROM `kraken-statistics`.statistics_torrentmeta
            WHERE date = @QUERYDATE));

END $$
DELIMITER ;
```

Appendix B

Sass/Compass update

Sass and Compass is directly installed as a gem in jRuby [5]. This jar file can be updated, if a new version of Sass or Compass is available. With the following step-by-step installation guide illustrates, how to update the Sass library used in the Kraken Web Interface. This guide was adapted, but initially provided by Xi Chen [16].

1. Go to http://www.jruby.org/download and download the latest *jruby-complete-[version].jar*.
2. Install the necessary gems locally:
   ```
   java -jar jruby-complete-[version].jar -S gem install -i ./compass-gems shared –no-rdoc –no-ri
   java -jar jruby-complete-[version].jar -S gem install -i ./compass-gems sass –no-rdoc –no-ri
   java -jar jruby-complete-[version].jar -S gem install -i ./compass-gems compass –no-rdoc –no-ri
   ```
3. Rename the jruby-complete.jar file to jcompass.jar
   ```
   mv jruby-complete-[version].jar jcompass.jar
   ```
4. Now compile all into one jar
   ```
   jar uf jcompass.jar -C compass-gems .
   ```
5. All done! Test it:
   ```
   java -jar jcompass.jar -S compass create –help
   java -jar jcompass.jar -S compass compile –help
   java -jar jcompass.jar -S compass watch –help
   ```

Appendix C

SUS

This appendix contains the standard SUS questions, which were asked during the usability evaluation questionnaire. The gathered data from the questionnaire and also the calculation of the SUS score is depicted in the preceding sections. It concludes with an approach to the user comments given in the questionnaire.

C.1 Standard Questions

The following questions were asked during the questionnaire of the usability evaluation. It contains all additional hints, which were given to the participants. The word cumbersome is explained in German, because a significant proportion of non-native English speaking people fails to understand this word [22].

1.	I think that I would like to use this system frequently. *Hint: Please answer this question from the point of view of a BitTorrent researcher.*
2.	I found the system unnecessarily complex.
3.	I thought the system was easy to use.
4.	I think that I would need the support of a technical person to be able to use this system.
5.	I found the various functions in this system were well integrated.
6.	I thought there was too much inconsistency in this system.
7.	I would imagine that most people would learn to use this system very quickly.
8.	I found the system very cumbersome to use. *Hint: According to http://dict.cc, cumbersome means: schwerfällig, mühselig, umständlich, mühsam.*
9.	I felt very confident using the system.
10.	I needed to learn a lot of things before I could get going with this system.

Table C.1: SUS Questions with hints like it was asked in the evaluation.

C.2 Response Data

	Date	CQ1	CQ2	CQ3	CQ4	CQ5	CQ6	SQ1-SQ10	Gender	Age Range	Highest Education
P1	17.10.2014	●	●	●	●	●	●	see SUS Calculation for SUS Responses	Female	18 - 25	Professional Maturity (Berufsmaturität)
P2	19.10.2014	●	●	●	●	●	◆		Male	26 - 35	Bachelor degree
P3	19.10.2014	●	●	●	●	●	●		Male	51 - 65	Postgraduate degree
P4	20.10.2014	●	●	◆	◆	●	▲		Male	26 - 35	Master degree
P5	20.10.2014	●	●	●	●	●	●		Male	26 - 35	Master degree
P6	20.10.2014	●	●	●	●	●	●		Female	18 - 25	Bachelor degree
P7	21.10.2014	●	●	●	●	●	●		Female	18 - 25	Bachelor degree
P8	22.10.2014	●	●	●	●	●	●		Male	18 - 25	Higher education entrance qualification (Maturität)
P9	22.10.2014	●	●	●	●	◆	●		Male	26 - 35	Bachelor degree
P10	22.10.2014	●	●	●	●	●	●		Male	26 - 35	Bachelor degree
P11	23.10.2014	●	●	●	●	●	◆		Male	26 - 35	Master degree
P12	23.10.2014	●	●	●	◆	●	▲		Male	26 - 35	Bachelor degree
P13	24.10.2014	●	●	●	●	●	●		Male	26 - 35	Bachelor degree
P14	24.10.2014	●	●	◆	●	●	●		Male	18 - 25	Bachelor degree
P15	26.10.2014	●	●	●	●	●	●		Female	51 - 65	Apprenticeship (Lehre)
P16	01.11.2014	●	●	●	●	●	●		Male	26 - 35	Master degree
P17	01.11.2014	●	●	●	●	●	●		Female	26 - 35	Bachelor degree
P18	05.11.2014	●	●	◆	●	●	●		Male	36 - 50	Master degree
P19	12.11.2014	●	●	●	●	●	◆		Female	18 - 25	Bachelor degree

CQ	Context Question
SQ	SUS Question

●	Correct Answer
▲	Missing Answer
◆	Wrong Answer

Table C.2: The responses of the users in the questionnaire.

C.3 Calculations

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
P1	5	1	9	2	10	2	9	1	9	2
P2	10	2	7	1	8	3	10	2	7	4
P3	8	3	6	3	8	5	8	1	8	1
P4	3	2	7	3	6	2	6	5	5	2
P5	10	1	8	10	9	1	10	1	9	1
P6	6	1	10	1	10	2	9	2	10	1
P7	10	1	9	1	10	1	10	2	7	1
P8	4	6	5	3	8	2	8	6	4	2
P9	2	8	2	3	2	8	6	8	3	3
P10	9	8	4	2	5	3	4	5	6	7
P11	8	2	8	1	9	1	8	2	9	2
P12	5	3	4	4	2	3	8	7	4	1
P13	8	2	3	1	9	2	10	2	8	1
P14	9	1	8	1	8	1	9	3	9	2
P15	5	2	2	4	9	5	7	3	5	10
P16	3	2	7	3	4	2	9	7	4	3
P17	3	7	3	5	4	3	4	8	8	3
P18	7	2	6	10	6	3	9	3	8	2
P19	7	4	6	3	8	3	8	4	6	3

Measured values from 1-10

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
2.8	1.0	4.6	1.4	5.0	1.4	4.6	1.0	4.6	1.4
5.0	1.4	3.7	1.0	4.1	1.9	5.0	1.4	3.7	2.3
4.1	1.9	3.2	1.9	4.1	2.8	4.1	1.0	4.1	1.0
1.9	1.4	3.7	1.9	3.2	1.4	3.2	2.8	2.8	1.4
5.0	1.0	4.1	5.0	4.6	1.0	5.0	1.0	4.6	1.0
3.2	1.0	5.0	1.0	5.0	1.4	4.6	1.4	5.0	1.0
5.0	1.0	4.6	1.0	5.0	1.0	5.0	1.4	3.7	1.0
2.3	3.2	2.8	1.9	4.1	1.4	4.1	3.2	2.3	1.4
1.4	4.1	1.4	1.9	1.4	4.1	3.2	4.1	1.9	1.9
4.6	4.1	2.3	1.4	2.8	1.9	2.3	2.8	3.2	3.7
4.1	1.4	4.1	1.0	4.6	1.0	4.1	1.4	4.6	1.4
2.8	1.9	2.3	2.3	1.4	1.9	4.1	3.7	2.3	1.0
4.1	1.4	1.9	1.0	4.6	1.4	5.0	1.4	4.1	1.0
4.6	1.0	4.1	1.0	4.1	1.0	4.6	1.9	4.6	1.4
2.8	1.4	1.4	2.3	4.6	2.8	3.7	1.9	2.8	5.0
1.9	1.4	3.7	1.9	2.3	1.4	4.6	3.7	2.3	1.9
1.9	3.7	1.9	2.8	2.3	1.9	2.3	4.1	4.1	1.9
3.7	1.4	3.2	5.0	3.2	1.9	4.6	1.9	4.1	1.4
3.7	2.3	3.2	1.9	4.1	1.9	4.1	2.3	3.2	1.9

Scaled-down values from 1-5

Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
1.8	4.0	3.6	3.6	4.0	3.6	3.6	4.0	3.6	3.6
4.0	3.6	2.7	4.0	3.1	3.1	4.0	3.6	2.7	2.7
3.1	3.1	2.2	3.1	3.1	2.2	3.1	4.0	3.1	4.0
0.9	3.6	2.7	3.1	2.2	3.6	2.2	2.2	1.8	3.6
4.0	4.0	3.1	0.0	3.6	4.0	4.0	4.0	3.6	4.0
2.2	4.0	4.0	4.0	4.0	3.6	3.6	3.6	4.0	4.0
4.0	4.0	3.6	4.0	4.0	4.0	4.0	3.6	2.7	4.0
1.3	1.8	1.8	3.1	3.1	3.6	3.1	1.8	1.3	3.6
0.4	0.9	0.4	3.1	0.4	0.9	2.2	0.9	0.9	3.1
3.6	0.9	1.3	3.6	1.8	3.1	1.3	2.2	2.2	1.3
3.1	3.6	3.1	4.0	3.6	4.0	3.1	3.6	3.6	3.6
1.8	3.1	1.3	2.7	0.4	3.1	3.1	1.3	1.3	4.0
3.1	3.6	0.9	4.0	3.6	3.6	4.0	3.6	3.1	4.0
3.6	4.0	3.1	4.0	3.1	4.0	3.6	3.1	3.6	3.6
1.8	3.6	0.4	2.7	3.6	2.2	2.7	3.1	1.8	0.0
0.9	3.6	2.7	3.1	1.3	3.6	3.6	1.3	1.3	3.1
0.9	1.3	0.9	2.2	1.3	3.1	1.3	0.9	3.1	3.1
2.7	3.6	2.2	0.0	2.2	3.1	3.6	3.1	3.1	3.6
2.7	2.7	2.2	3.1	3.1	3.1	3.1	2.7	2.2	3.1

SUS algorithm applied

Multiplied sum
87.7778
83.3333
77.7778
64.4444
85.5556
92.2222
94.4444
61.1111
33.3333
53.3333
87.7778
55.5556
83.3333
88.8889
54.4444
61.1111
45.5556
67.7778
70.0000
Ø 70.9357

Table C.3: The calculations made during the evaluation of the questionnaire.

C.4 Approaching the User Comments

In the evaluation, there was one optional question at the end of the questionnaire, which asked for a comment. This question was commonly used and several issues were mentioned. This list discusses the mentioned issues:

The page jumps sometimes to the top of the page.
The Kraken Web Interface is mobile optimized. Therefore it needs to recalculate its map on every change of the browser window width. The map is redrawn and then the browser is scrolled to the top of the map. If the map is not the main concern of a page, this feature could be obstructive.

The menu was not intrusive enough and the specific information were not found.
The menu has three different buttons for the three main sections of the page: General & Specific information and System Status. The torrent list and the detailed map is together behind one button in the menu, because they belong to each other. The detailed page cannot be opened without specifying what movie should show up. For further work, those two pages could be integrated into one page to resolve this problem.

The top menu "Torrent Visualization" is also selected on the details page.
This problem is similar to the aforementioned. The menu "Torrent Visualization" congregates all the specific information, which means, the torrent list and the detailed page. It could be integrated into one page with one menu button in further work.

The search bar and the torrent list disappears after the selection of a torrent.
Currently, the torrent list and the details page are two different pages. For this reason, the search bar and the torrent list disappear when selecting a movie. By integrating these two pages into one page, this issue could be resolved.

The navigation on the page is not straight-forward. A back button is missing and the user always needs to navigate with the top menu.
The back button is not inside the page, but the behavior of the browser back button works as expected. If the user returns with the browser back button from the details page to the torrent list, he receives the same page he left before. This means, that the search and the selected date are the same as before. If the user copies the detailed page URL, he will return to the exact same page. If a date is selected, it is also stored in the address line. For further work, a back button could be integrated into the details page. It could be linked to the standard back button command of the browser.

The user has to click too many times before he gets the relevant information (especially the CSV download).
The Kraken Web Interface is a website to present and spread the scientific work on Kraken. The data can be downloaded and further processed by researchers by downloading the data in the CSV format. The main goal of the Web Interface is not to

induce researchers to use the raw data, but to get to know to work behind Kraken. If they are interested in expanding the work on Kraken or to participate in further research, contacting the Kraken researchers would be the way to go.

The entries on the torrent list can not be filtered by country.
This is the case by design. Not every movie can be listed when opening the torrent list, because its size would be too long. Therefore, a filtering needs to be done. At the moment, every observed torrent on one day appears when selecting a date in torrent list filter. This filtering could be improved by handling more than a keyword and a date. One possibility to improve the torrent list is to eliminate the date selection and adding an endless scrolling. Endless scrolling means, that a new chunk of data is reloaded when the user scrolled to the bottom of the page. This would resolve the issue with the day selection. If the list is then ordered by date and searchable with keywords, the usability would probably improve.

It is expected to choose the movie before the day of observation (torrent list).
This issue could be resolved by implementing an endless scrolling as mentioned previously.

The search displays too many entries and the matching title not on the top.
Currently, the search of the torrent list is splitting up the keywords on a space and shows all entries where at least one keyword is present. If the user searches for single-lettered or short keywords, every movie with this letter or keyword shows up. A search for ”a new house” leads to a search result of every movie with an ”a” in it. The search results are not reordered again, the search only hides non-matching movie entries. This leads to the situation, that an exact match of the keyword and one torrent title may not show this movie on the top, if there are is more than one possible match. In further work, there is room for improvement here.

The map (especially the generic one) should not be restricted to one day. It is not straight-forward, why a date needs to be selected.
This restriction was introduced to keep the logic of the map simple in this first version of the Kraken Web Interface. If there is more data to show on the map than from one day, a day range selection would be necessary. This would need boundaries of the lifespan of the torrent (or its observation). Furthermore, if on one day no peer was observed (by technical problems or just because nobody downloaded the movie) and on the following days, the peers were starting to download again, the range would need to be capable of handling gaps. These problems do not occur, when one day needs to be selected. Though, an extension of the Kraken Web Interface could implement this feature.

With a broad dataset, it is possible, that the peak of the curve is not on the right day in the timeline graph of the details page.
The timeline graph on the details page is a spline chart and has therefore rounded curves. With a broad dataset, this involves, that the peak of a graph is not always on the highest data point, but slightly next to it. For a more precise depiction of the time graph, it would be necessary to switch from a spline chart to a line chart with sharp edges by simply changing one property. C3.js supports both kind of graphs. In further work, the benefits and disadvantages could be discussed.

Appendix D

Installation Guidelines

The installation of Kraken Web Interface was tested in Eclipse with a number of additional plug-ins. It cannot be guaranteed, that all the steps work or are similiar in different IDE's. If you follow the steps given, you can run the Kraken Web Interface locally on a Tomcat Server.

D.1 Development Environment

To further develop on the Kraken Web Interface, you can follow these steps to set up the environment.

1. Download a new instance of ”Eclipse IDE for Java EE Developers” from the official website http://www.eclipse.org.

2. Go to http://eclipse.org/subversive and drag and drop the install button into Eclipse. Install Subversive SVN.

3. Go to http://eclipse.org/m2e and copy the Maven Eclipse Plugin Repository Path. Install the Eclipse Maven Plugin by selecting Help - Install new Software and then pasting the link. It may already be installed.

4. Go to http://www.aptana.com and download and install the Eclipse Plugin-in of Aptana Studio 3.x.x similar to the Eclipse Maven plug-in. This plug-in is necessary for the syntax highlighting in SCSS (Sass) files.

5. Go to http://github.eclipsesource.com/jshint-eclipse/ and drag and drop the install button into eclipse. Install the JSHint plug-in.

6. Go to http://tomcat.apache.org and download Tomcat 8.x.x in a zip version. Unpack and move it to a directory of your choice.

7. Open the ”SVN Repository” perspective (Window - Open Perspective - SVN Repository) and add a new Repository Location with this URL:
   https://svn.csg.uzh.ch/svn/kraken/trunk
   Use also your SVN credentials here.
8. Check out at all kraken-* projects, JKad_Development and TTorrent. Relevant for the development is only kraken-ui, but there are many dependencies between the projects.
9. Convert all projects, which are not yet Maven projects into Maven projects: Right click on a project - Configure - Convert to Maven Project.
10. Open the Servers view in Eclipse (Window - Show View - Others - Servers) and add a new Tomcat 8 server. You need to link it to the download/installation directory of Tomcat 8. Add the kraken-ui project to the newly created server.
11. Run Maven install on kraken-ui. The source files are now generated.
12. Connect to the University of Zurich with a VPN connection or be sure to be connected to a WiFi hotspot inside the University of Zurich. This is necessary to connect to the remote database.
13. Start the server in the servers view.

D.2 Build process

- The minification is only enabled in the productive build, in debug mode it does not minify.
- Debug build: Run the build without adding a build profile. Debug mode is standard.
- Productive build: Add a new Maven build configuration with the goal ”install”. Add the build profile ”productive”. The source files will now be minified.
- Deployment: To deploy, you can add the server address in the pom.xml file. If you then run a Maven build process with goal: ”tomcat:redeploy” it automatically deploys to the specified server. You still need to add the productive profile.

D.3 Build from CD

1. Do steps 1 to 6 from Section D.1.
2. Copy the contents from the folder /source on the CD to your workspace directory.
3. Import the projects with File - Import.
4. Do steps 9 to 13 from Section D.1.

Appendix E

Contents of the CD

/

You can find three different files in the root of the CD. The abstract in English, called *Abstract.txt*, the abstract in German, called *Zusfsg.txt* and this Bachelor Thesis in the PDF format, called *Bachelorarbeit.pdf*.

/source

The *source* folder contains all source files from the Kraken Core software and Kraken Web Interface. The files can be copied to a set up an Eclipse instance and to build the Kraken Web Interface project. Please take a look at Appendix D.3 to see how the Kraken Web Interface can be runned on a local server.

/database

In the *database* folder, you can find an SQL script for the set up of the database schema, used by the Kraken Web Interface. The script, which automatically aggregates the data from the Kraken Core database to the Kraken Web Interface on a daily basis is also in this folder.

/latex

In the *latex* folder, you can find a LaTeX version of this Bachelor Thesis. All images, text and source files are included.