Metacommunication

Diplomarbeit

Zur Erlangung des akademischen Grades

Master of Science in Engineering (MSc)

der Fachhochschule Campus Wien
Masterstudiengang IT-Security

Vorgelegt von:
Ing. Gerald Weber, MSc

Personenkennzeichen
1210537015

Erstbegutachter/in:
Martin Mulazzani, PhD

Zweitbegutachter/in:
Markus Huber, PhD

Eingereicht am:
19.05.2014

Erklärung:

Ich erkläre, dass die vorliegende Diplomarbeit von mir selbst verfasst wurde und ich keine anderen als die angeführten Behelfe verwendet bzw. mich auch sonst keiner unerlaubter Hilfe bedient habe.
Ich versichere, dass ich dieses Diplomarbeitsthema bisher weder im In- noch im Ausland (einer Beurteilerin/einem Beurteiler zur Begutachtung) in irgendeiner Form als Prüfungsarbeit vorgelegt habe.
Weiters versichere ich, dass die von mir eingereichten Exemplare (ausgedruckt und elektronisch) identisch sind.

Datum: Unterschrift:

Kurzfassung

In den vergangenen Jahren trat die Vorratsdatenspeicherung in allen Mitgliedsstaaten der Europäischen Union in Kraft und beeinflusst unser tägliches Leben. Egal in welcher Ausprägung Vorratsdatenspeicherung in den Mitgliedsstaaten umgesetzt wurde, es handelt sich immer um die Speicherung von Spuren - sogenannten Metadaten - die jede Person in dieser digitalen Welt hinterlässt. Die Politik sieht keine Auswirkungen auf die Privatsphäre von Einzelpersonen und legitimiert die Speicherung durch Terrorismusbekämpfung. Weil die Richtlinie der EU keine Vorgaben zur Datenspeicherung der Vorratsdaten vorsieht, dürfen diese sensiblen Daten auch außerhalb der Mitgliedsstaaten gespeichert werden. Man kann also davon ausgehen, dass Regierungen, Unternehmen sowie Nachrichtendienste (Stichworte: Edward Snowden, COMINT und PRISM) Zugriff auf diese Daten haben und Profile von Individuen, basierend auf dem Kommunikationsverhalten, anlegen. Zur Sensibilisierung der Öffentlichkeit, versucht diese Arbeit die vorhandenen Daten von Benutzern aufzubereiten und zu visualisieren.

Abstract

In recent years, data retention has come into effect in the member states of the European Union and affects our daily life. Data retention, in which occurrence whatsoever deals with the recording and storage of traces - called metadata - everyone leaves behind in this digital world. Public authorities argue, that these records are essential for suppression of terrorism and has no influence on privacy of individuals. While the EU failed to dictate specifications to ensure data security, these sensible metadata can be stored everywhere, even outside of the EU. Nevertheless, governments and companies as well as intelligence agencies (keywords: Edward Snowden, COMINT and PRISM) have access to these information and can build *a pattern of life* based on the communication habits of individuals. To sensibilise citizens of the European Union, this work tries to visualise these patterns of communication.

Contents

List of Figures

List of Tables

Listings

Chapter 1

Introduction

1.1 Introduction

According to Directive 2006/24/EC of the European Parliament and the Council (see [14]), European Member States are obliged to implement data retention. This forces telecommunication and internet service providers (TSP/ISP) to store call detail records as well as internet traffic and transaction data over a duration of six months up to two years. The concerned data is called metadata and includes amongst others identification of calling and called party, the starting time, call duration as well as location data. Governments can use the data to localise individuals, analyse habits and their social network to circumvent terrorist attacks. Organisations store these data too, but claim to use this information only to optimize their services. Nevertheless, due to publications of Edward Snowden[1], which raised awareness for encrypted communication, individuals turn their attention towards privacy protection and renders third parties (hopefully) unable to analyse the content of communication by using secure channels. So, governments and organizations invest a lot to analyse individuals based on the available metadata. Metadata, also called the data of data, allows intelligence agencies to analyse communication and movement patterns to create detailed profiles of individuals. While metadata occur every time individuals communicate digitally and shed light on personal communication behaviour we refer communication metadata as metacommunication.

The worlds largest intelligence organisation is the United States National Intelligence Agency (NSA) head-quartered in Fort Meade. Founded in 1952, succeeding the Armed Forces Security Agency (AFSA)[15], the NSA is responsible for the collection and analysis of global communication. These activities are also known as signals intelligence (SIGINT) and describes the interception of signals (e.g. by eavesdropping). It is separated into human communications intelligence (COMINT) and electronics intelligence (ELINT) which concentrates on non-communication signals (e.g. radars). COMINT deals with the interception of foreign, human communication. Recent disclosures of secret documents by Snowden revealed, that the NSA is not only recording foreign communication when passing or terminating in the U.S., but also the communication of allies and U.S. citizens. The agency argues, that these actions are consti-

[1] Whistle-blower and former NSA contractor

tutional and essential to protect the public from terrorist attacks. The U.S. public on the other hand is afraid of mass surveillance and can not believe, that the NSA is allowed to build a *pattern of life* of individuals and their social network three hops in depth (Pool and Kochen 1978 investigated the *small world* principle with 500 persons knowing 500 others knowing 500 further persons resulting in a small world of 12.5 million people[16]). Another discovery is, that at least 9 American internet companies are collaborating[17] (voluntarily or not[2]) with the NSA (e.g. Microsoft, Google, Yahoo, Facebook, Apple...) as well as Verizon[3]. Verizon has to deliver telephony metadata to the NSA on a daily basis (see [18]). Recently, Snowden leaked documents, showing that the NSA has created a surveillance system that is able to record all phone calls of a foreign nation and play back these conversations up to 30 days later like a time machine. So it is not only possible to collect metadata of telephone calls, but it is technically possible to eavesdrop all telephone conversations of a whole country.

Nevertheless, the European Court of Justice declared the Data Retention Directive to be invalid in April 2014 (see [19]). This again forces European member states to adapt their national regulations.

Based on the fact, that at least the provider company has access to communication metadata, this paper aims at visualising these information and the social network of persons only by the usage of the provided communication metadata. The individual should be brought to eye level with their TSP/ISP and see what analysis is possible with the stored data. A constraint of this paper is that it has only access to the data provided by one user and not a network of users interacting with each other. Therefore only unidirectional communication data can be analysed and visually presented.

The research questions are in detail:

- How could an application be designed to guarantee privacy to users uploading personal data for visualisation purposes?
- Is it possible to decrypt Whatsapp's (see section 2.6.1) message database and use the content to visualise a social network?
- What are the benefits of a stream-graph visualisation of telecommunication data?

Chapter 2 describes related work and provides additional information to metadata and social network theory. Chapter 3 identifies the most appropriate application design, whereas Chapter 4 describes the implementation process of the project in detail. After the implementation, chapter 5 presents the final application. Finally, Chapter 6 concludes and attempts to give an outlook to future work.

[2]The NSA program is called PRISM; Information gathered and leaked by Edward Snowden, published by The Guardian/Glenn Greenwald

[3]American broadband and telecommunications company

Chapter 2

Background

This section discusses the most relevant topics for this thesis.

2.1 Related Work

The first referenced project called **Immerson** presents the user *"a people-centric view of your email life"*[4]. Developed by the MIT Media Lab, this project concentrates on the analysis of email communication. The user has to provide credentials for a supported email service (Gmail, Yahoo, Exchange) and Immerson starts analysing metadata like from, to, cc and time-stamp. Based on this information, it walks through the social network and visually presents it. The authors assures secure storage of the collected data and claims that Immerson is about self-reflection, art, privacy and strategy. So it provides the user with the information with whom data is shared and presents different perspectives of their email interactions.

Prismviz [5] has a similar approach but is based on the analysis of SMS communication. The required data is supplied in form of iPhone backups generated by iTunes and is visualised as a stacked-graph [7]. Prismviz, developed by Bay Gross provides a comprehensive user interface which allows the user to sort, colour, blend, and mix the SMS history. Additionally, the provider claims only to extract metadata and not to save any data beyond the user's session.

Finally, **MetaPhone** [20], a Stanford project developed by Jonathan Mayer provides an Android mobile phone application, which logs the user's call and text usage and tries to analyse the behaviour by using public sources like Yelp[1] and Google Places. By using these additional resources, MetaPhone is able to cumulate information to reveal the users activities, traits and relationships. The NSA confirmed collecting American phone records, but argue that it has little privacy impact. That may be correct in case someone looks at each call individually. When a person starts to call a number regularly, which is a support hotline for a special pharmaceutical product, it is possible to conclude, that this person is affected by this illness. Unfortunately, the author does not

[1] **What is Yelp?** Yelp is an online urban city guide that helps people find cool places to eat, shop, drink, relax and play, based on the informed opinions of a vibrant and active community of locals in the know

provide any graphical representation of the concluded information. The MetaPhone project collected information about hundreds of users over a period of several month, whereas the NSA has collected data from millions of Americans over several years, growing daily. An in-depth analysis could reveal much more sensitive information than MetaPhone was able to reveal.

2.2 Metadata

This section describes what metadata is and what it can be used for.

In the context of this paper, the term metadata (also termed as context data) refers to the part of a communication, that describes the technical parts of a conversation. The content of a conversation or email is not part of metadata. The Data Retention Directive 2006/24/EC lists *"Categories of data to be retained"* [14] in Article 5 which contains:

- data necessary to trace and identify the source of a communication (telephony and internet access)
- data necessary to identify the destination of a communication
- date, time, duration of communication
- type of communication
- identification of users telecommunication equipment
- location of mobile equipment.

The adaptation of the Austrian Telecommunications Act 2003 with the Federal Law Gazette[2] I No. 27/2011 §102a (see [21]) concretise the requirements for TSP/ISP.

Recent disclosures of intelligence documents could imply, that the content of a conversation is already available for some agencies too, but that is (hopefully) an assumption. Users can prevent or make it harder for others to read private communication by using modern encryption algorithms with sufficient key lengths (see recommendations by NIST[3][22]). Bruce Schneier[4] still believes in the mathematics, even though the NSA made a breakthrough in cryptoanalysis[23]. Nevertheless, this work restricts its analysis to the metadata which is freely available to customers of TSP/ISP. These companies allow customers to download usage information in CSV[5] file format or PDF[6]. Depending on the provider, these files looks different but contains the following information:

- source (only outgoing communication, therefore the registered customer)
- start of the communication (date, time)

[2]German: Bundesgesetzblatt
[3]National Institute of Standards and Technology
[4]American cryptographer and security specialist
[5]comma-separated values
[6]Portable Document Format

- duration (precise to the second or data volume in kilobytes for data services)
- service (telephony, SMS/MMS[7], data services[8])
- target (e.g. another customer or the service provider itself in case of data services).

2.3 Privacy

This section has a look on privacy impacts depending on the collection and processing of communication data and especially on metadata. The points of view are grouped into legal, technological and economical.

2.3.1 Legal view

The legal view will explore the European and the U.S. privacy regulations. The European, while these are immediately effective for the authors of the thesis, the U.S. while recent revelations of Edward Snowden have put these into global spotlight.

While many countries recognises a right of privacy explicitly in their constitution, these provisions at a minimum include rights of inviolability of the home and secrecy of communications. These constitutions do not differentiate between analogous and digital communication due to their age nor do these specify rights to control one's personal information held by the government or third parties. One exception is the relatively young constitution of South Africa (promulgated 1996) describing in its Chapter 2 Article 32 the rights to "access to information".

European regulations

The European Union passed the directive 95/46/EC[24] because of the shortcomings and differences of the implemented law of its member states. This directive on the *"protection of individuals with regard to the processing of personal data and on the free movement of such data"*[24] raised the bar for national law. Members states had to implement the directive into national law by October 1998 to protect the rights and freedoms of persons regarding to the processing of personal data. It contained regulations about the quality of personal data, the legitimacy of data processing, information obligations, access rights of the affected persons as well as exemptions, restrictions and confidentiality rules. But, shortly following the Madrid bombings on 11th March 2004, the European Council released the "Declaration on Combating Terrorism"[9] which contained proposals to simplify information exchange (including personal information like DNA, fingerprints and visa data), retention of communication traffic data, obligation for carriers to communicate passenger data (passenger name record [PNR]) as well as

[7]short message service/multimedia message service

[8]Data services refers to communication technologies like GPRS, EDGE, UMTS, LTE

[9]European Council, Declaration on Combating Terrorism, `http://www.consilium.europa.eu/uedocs/cms_data/docs/pressdata/en/ec/79637.pdf`, last seen on 10th July 2014

the establishment of surveillance mechanisms. The proposals were affirmed after the London bombings on 7th July 2005 and concluded into the Directive 2006/24/EC (the Data Retention directive)[14], which denotes a shift from targeted to mass surveillance. After publication of this directive, member states had time to adopt national law to execute data retention. Austria adopted the Austrian Telecommunications Act 2003 (see Federal Law Gazette 2 I No. 27/2011 §102a [21]) which went into action on 1st April 2012. Germany adopted the Act for the Amendment of Telecommunications Surveillance on 21st December 2007 but was declared unconstitutional by the Federal Constitutional Court on 2nd March 2010. Interim development, a proposal for a General Data Protection Regulation[10] discusses a framework for the protection and free movement of personal data. The key changes of this regulation would include:

- The right to be forgotten (deletion of data without legitimate grounds for retaining it).
- Explicit consent for data processing.
- The same set of rules for all European Union member states.
- Increased responsibilities for organisations processing personal data.
- EU rules apply also to organisations offering goods or services in the EU or monitor the online behaviour of Europeans.

Nevertheless, on 8th April 2014, the Court of Justice of the European Union declared the Directive 2006/24/EC invalid. So, all member states, that implemented the directive have to repair their national law to be conform with the latest changes.

Another surveillance method called *behavioural targeting "is the monitoring of people's online behaviour over time"*[25] to present people specific advertisements. Tracking users by the usage of cookies, supercookies[11], device fingerprints and deep packet inspection is common for commercial websites. The European Data Protection Directive 95/46/EC[24] gives users several rights which are triggered in case any company processes personal data. Even information without a name[12](e.g. IP Address held by ISP) as well as information, that can distinguish a person within a group are personal data. Therefore, *"in most cases, data protection law applies to behavioural targeting"*[25]. The European Union's laws want to guarantee transparent data collection and processing (transparency principle). Companies have to inform users that information is collected and have to *"obtain the user's consent before storing or accessing cookies on a device"*[25]. For functional cookies (e.g. login status, shopping cart) no consent is required. In this way, the EU wants to invent a Do-Not-Track-Standard for all browsers and websites. A final note on the U.S. behavioural targeting: While European websites needs consent to begin tracking activities, U.S. websites requires an explicit opt-out to stop tracking.

[10]European Commission, COM 2012/11 final, `http://ec.europa.eu/justice/data-protection/document/review2012/com_2012_11_en.pdf`, last seen on 10th July 2014

[11]aka Flash cookies or local shared objects; Flash cookies are capable of storing information about the settings and circumvent the user's preferences

[12]Court of Justice of the European Union, 24 Nov. 2011, Case C70/10 (Scarlet/Sabam), par. 51

U.S. regulations

The U.S. Constitution[13] grants (in the Fourth Amendment) the right, that without a warrant, people shall be secure against unreasonable searches and seizures. But since the invention of the telephone, wiretapping is used by the police to investigate crimes invisibly without access to a suspect's property. The first court decision in 1928 (Olmstead case[14]) allowed wiretapping *"operations of the federal prohibition agents were not a search and seizure in violation of the security of the persons, houses, papers and effects of the petitioners in the constitutional sense"*[26]. The following Federal Communications Act rendered interception and disclosure of wired communications illegal. Nevertheless, the Federal Bureau of Investigation (FBI) carried on wiretapping because they interpreted interception allowed as long the communication is not disclosed outside the FBI. Wiretapping without court orders went on, until the Katz decision 1967 reinforced that *"the Fourth Amendment protects people, not places"*[27] which requires a warrant. Additionally, Title III of the *Omnibus Crime Control and Safe Streets Act* raised the bar for warrants regarding electronic searches in 1968. The Foreign Intelligence Surveillance Act (FISA) 1978 allowed wiretaps when

- the subject is believed to be involved in illegal activity and
- the subject is an agent of a foreign power or terrorist group.

In 1986, the Electronic Communications Privacy Act (ECPA) regularised real-time capture of calling data - so called pen register and trap and trace. Also call detail records (CDR) which contains numbers, time and duration were captured. While telephones were stationary and these CDRs were shared with telephone companies, Constitutional protections did not affect these. With technological development, investment in U.S. communications infrastructure and the rise of IP-based traffic over fibre optical cables, it became more difficult to differ between transit and domestic traffic. For domestic traffic, a warrant was still required. To conduct wiretapping, the Communications Assistance for Law Enforcement Act (CALEA) forced infrastructure companies to place a backdoor in their equipment to allow the FBI eavesdropping of communication. After the attacks on 11th September 2001 warrantless interception of reasonably believed transit traffic was demanded. Installations of secret rooms at network infrastructure nodes to eavesdrop the internet traffic (e.g. AT&T Room 641A in San Francisco[15]) and cooperation of service providers and internet companies shows, that not only legal action took place. While wiretapping must be supervised by the FISA court, the *Protect America Act of 2007* allows warrantless action of communications that begin or end in a foreign country without supervision. Ultimately, the FISA Amendments Act 2008 grants immunity to telecommunication companies which delivered CDRs to the NSA.

[13] The U.S. Constitution, `http://constitutionus.com/`

[14] Olmstead vs. United States, `http://www.law.cornell.edu/supremecourt/text/277/438`, last seen on 10th July 2014

[15] The secrets of Room 641A, `http://www.zerohedge.com/sites/default/files/images/user5/imageroot/2013/06/NSA%20wiretaps.pdf`, last seen on 10th July 2014

The NSA argues, that dozens of plots could be stopped with the help of surveillance activities[28]. But, the three briefly mentioned cases did not justify the expenses and the illegal actions of the eavesdropper[29].

2.3.2 Technological view

FBI claimed, that switching technology reduce the effectiveness of wiretapping and achieved a tremendous success with the Communications Assistance for Law Enforcement Act 1994 (CALEA). CALEA forces telephone companies to build their equipment wiretapping ready. These modifications created risks that were exploited in Athens 2005, when an investigation found out, that *"for 10 months someone had been wiretapping senior members of the Greek government"*[30]. CALEA only applies to the U.S., but manufacturers try to have as few versions of their equipment as possible. The ordered Ericsson switch came with wiretapping software present but supposed to be shut off and in return, the auditing software for wiretapping was not present. So, unknown parties enabled wiretapping features unseen and started eavesdropping communication in Athen.
Another surprise were the revelations of the AT&T technician Mark Klein, making Room 641A public where the NSA had access to foreign and domestic traffic.

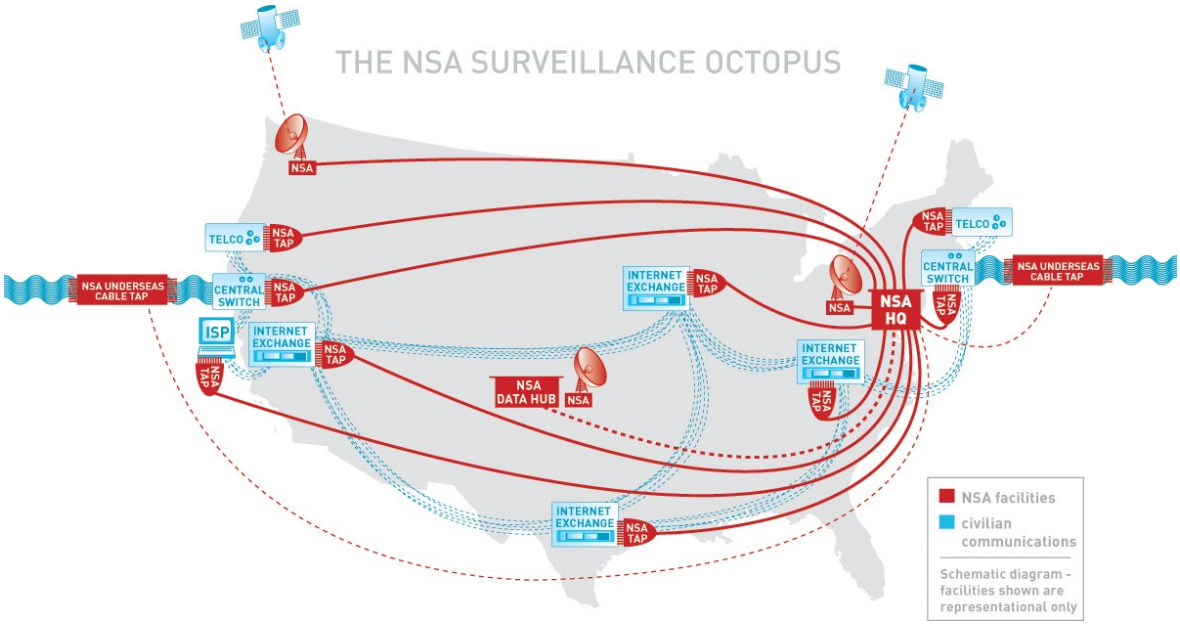


Figure 2.1: The NSA surveillance octopus[1]

While it is not possible to store the complete traffic, only CDRs as well as internet connection establishments are stored. Additionally, in our mobile world, people are linked with their mobile telephones. These devices allows precise tracking of personal movement like the German Green party politician Malte Spitz rigorously showed. He sued Deutsche Telekom to hand over 6 months of his phone data. This information were

combined with Spitz's agenda at this time and visualised online[16]. So even without knowing the content of a communication, a lot of information is revealed by metadata, which is automatically created and stored by the TSP/ISP. Furthermore, it is much easier to search huge amounts of metadata than listening to millions of phone calls. Even a former general counsel of the NSA, Stewart Baker said, that metadata tells you everything about someone's life. *"If you have enough metadata you don't really need content... It's sort of embarrassing how predictable we are as human beings"*[31]. Additionally, former director of the NSA and the CIA, General Michael Hayden confirms and asserts further, that *"we kill people based on metadata"*[17][32].

Another technological influence on privacy comes from the NSA attempts to weaken cryptographic implementations by compromising a National Institute of Standards and Technology (NIST) cryptographic standards[33]. These standards are adopted by the industry and built into communication hardware and software allowing the NSA to analyse encrypted communication.

Based on behavioural targeting (see section 2.3.1), scientists argue, that the concerns about NSA programs are overreaction. A network scientist working for a telephone company sees no difference in identifying key customers based on metadata in comparison to *"unmasking the leaders of terrorist networks"*[34]. Others are not worried with the current government, but afraid of future governments using these information against regime enemies. Finally, *"we have the technology and we have the data. We have to use it. So simple is it"*[34].

Nevertheless, gathering information by wiretapping is threatening. While the CALEA solution is based on the fact, that communication was centralised, new link routing technologies were invented without requiring a centralised design. So the complexity and effort for wiretapping communication raised. Additionally, adding wiretapping interfaces to devices create new vulnerabilities. As the Greek incident proofed, *"experience shows that if a vulnerability exists in a security system, it is likely that someone will take advantage of it sooner or later"*[35]. Therefore, the implementation of wiretapping interfaces and control systems into communication infrastructures poses serious risks[30]. Due to the importance of the internet to private and public computing, communications security should have absolute priority over communications surveillance[30]. A solution for this surveillance problem could be the exploitation of naturally occurring bugs in communication end devices[36]. To succeed in this field, the FBI requested US$15 million and established a Domestic Communications Assistance Center (DCAC) in 2012. So instead of introducing vulnerabilities in the network infrastructure, law enforcement could concentrate on exploiting existing weaknesses. This would lead to higher communication security, which protects private and public systems from criminal exploitation and enables law enforcement to conduct legally authorised wiretapping. A win-win situation for society and law enforcement.

[16] Zeit online, Tell-all telephone, `http://www.zeit.de/datenschutz/malte-spitz-data-retention/`, last seen on 10th July 2014

[17] The Johns Hopkins Foreign Affairs Symposium Presents: The Price of Privacy: Re-Evaluating the NSA; `https://www.youtube.com/watch?v=kV2HDM86XgI`, last seen on 10th July 2014

2.3.3 Economical view

A look at commercial services reveals, that mobile telephone carriers earn a lot of money by selling their users location data to third parties[37]. Based on the legal (see section 2.3.1) and technological (see section 2.3.2) aspect of privacy, it is obvious, that U.S. law enforcement and intelligence agencies collect information from commercial services (e.g. AT&T secret room, Verizon CDRs, data from internet companies). So, the authors of [12] looked at the size, costs and power consumption of datacenters (see table 2.1):

Company	Costs	Estimated storage	Power
Amazon (for CIA)	US$600 mio.	900 Petabytes	
Bloomberg	US$710 mio.		
Microsoft	US$1000 mio.	300 Petabytes	200 MW
NSA	US$2000 mio.	3-12 Exabytes[18]	65 MW
Facebook (one of many)	US$1500 mio.	100 Petabytes	78 MW
Google (one of 8)	US$600 mio.	15 Exabytes[19]	220 MW

Table 2.1: Overview of datacenters[12]

Another business based on user data is the direct-mail industry, which has revenues of US$11 billion. The mailing-list provider, Acxiom has approximately 500 million people records occupying 12 Petabytes. Equifax, a main credit recording service handles 26 Petabytes of data. So, companies engaged in selling data create high revenues. The online tool called Swipe[20] calculates the worth of information pieces concerning persons. The more meaningful an information for the credit service is the higher is its worth. While private data collection has only few constraints in the U.S., large accumulation were created. The main problems with this information is, that it is hard to proof its correctness and even harder to correct it. Furthermore, once collected information will have an impact on your further live and generate revenues for companies you have never heard from.

2.3.4 Future directions

The privacy issues of data retention needs new solutions for balancing law enforcement requirements and privacy assurance. Some approaches exists to address security threats, social principles and security requirements with a combination of encryption and data integrity, access controls as well as digital signatures, logging and description of handover interfaces[38]. After the revelations of Edward Snowden, the public aroused and will (hopefully) pay more attention on changes of privacy regulations.

[20]Swipe - calculate the value of information, `http://turbulence.org/Works/swipe/index.html`, last seen on 10th July 2014

2.4 Social Network Analysis

This section will provide a short introduction to social networks and analysis of social networks.

2.4.1 History

The idea of seeing groups of people as a social network has its roots in the sociological tradition. In the 1930s, based on the work of Simmel (Fundamental Questions of Sociology, 1917) and others, researchers began to investigate social relations from a technical view. They used the terms 'points', 'lines' and 'connections' to describe patterns of social relations. Influenced by these formal ideas, Bott (Observation of play activities in a nursery school, 1928), Moreno (Who shall survive, 1934) and Lewin (Principles of Topological Psychology, 1936) published their works in the field of social relations characterizing it as a network. Moreno introduced the term *sociogram* for a graph containing persons and relations where mathematicians only see vertices and edges[39]. Barnes (1954) suggested to take the metaphor of the network serious, whereas Bott (1955, 1956) applied measures of connectedness and density. Based on the work of Lévi-Strauss and Weil (1949), White (1963) initially applied algebra to represent social structures. *"White himself worked with others on algebraic methods for representing and analysing systems of social positions and roles"*[40] which resulted in a new generation of social network researchers. Until the introduction of graph theory into the field of social analysis, analysis of large sociograms was complex and tedious[41]. To describe the power of banks in the corporate world of America, Bearden et al. (1975) introduced the centrality measure.

Another important field of application was covered by Warner, who investigated community structures. *"Fischer (1977) and Wellman (1979) generated work that completely reoriented the research area"*[40].

The most important development in social network analysis has been growing interest from outside the social sciences with Milgram's work on 'small worlds' (1967) and random networks. By ignoring previous work on social networks, physicists applied these ideas on the social world with great success. While collecting information about social networks was tedious, the advent of interactive Web 2.0 technologies simplified the acquisition to relevant and large databases and therefore accelerated the development in the field of social network analysis.

"Since its 'take-off' in the 1970s, the volume of published research on social networks has grown exponentially (Knoke and Young, 2008: 1-2), while the number of subject areas in which it is being employed has experienced almost linear growth, from a handful to almost 60 by the year 1999 (Freeman, 2004: 5)"[40].

2.4.2 What is a Social Network?

In graph theory, *"a network is, in its simplest form a collection of points joined together in pairs by lines"*[2]. The points are called **vertices**[21] or **nodes** whereas the lines are

[21] Singular: vertex.

called **edges**.

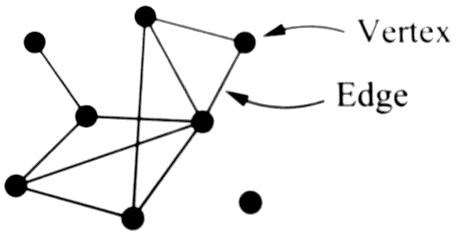


Figure 2.2: A small network composed of eight vertices and ten edges [2].

The nodes in a social network can be persons, organizations, web pages or countries and generally depends on the field of research whereas the edges are relations between them. Analysis of these networks is based on the assumption that social life primarily relies on relations and the patterns formed by these relations.

Depending on the research target, selecting the correct nodes poses an early challenge. So Laumann et al.[42] determined three possibilities to address the boundary specification problem:

1. The position-based approach only pays attention to formally defined network members - e.g. employees in a organizational department.

2. The event-based approach defines relevant events and select network members according to their attendance.

3. The relation-based approach starts with a small set of prospective members and expands the network by including nodes sharing particular relations to members.

After application of these non-exclusive approaches for setting up a network for an analysis, relevant relations between the nodes have to be identified. Due to the vast number of different relations (e.g. collaborations, friends, web links, information flows, ...) Borgatti[3] reduced them to four groups (see figure 2.3):

1. Similarities (e.g. attitudes, locations, group memberships, ...)

2. Social relations (e.g. kinship, friend, fellow student, ...)

3. Interactions (e.g. speaking with, helping, inviting someone, ...)

4. Flows (e.g. resource or information flow, ...)

Additionally, social network analysis take context seriously. Not (only) individual attributes makes nodes behave in a particular way, but their relation with other nodes influences them. *"Social network analysts argue that causation is not located in the individual, but in the social structure"*[40]. So people does not solely behave similar because they have similar characteristics, but also because their common characteristics put them into identical social positions within a network. An social network is not only the sum of individual behaviours, but acting of individuals on one another *"to shape one*

Similarities			Social Relations				Interactions	Flows
Location	Membership	Attribute	Kinship	Other role	Affective	Cognitive	e.g.,	e.g.,
e.g.,	e.g.,	e.g.,	e.g.,	e.g.,	e.g.,	e.g.,	Sex with	Information
Same spatial and temporal space	Same clubs	Same gender	Mother of	Friend of	Likes	Knows	Talked to	Beliefs
	Same events	Same attitude	Sibling of	Boss of	Hates	Knows about	Advice to	Personnel
	etc.	etc.		Student of	etc.	Sees as happy	Helped	Resources
				Competitor of		etc.	Harmed	etc.
							etc.	

Figure 2.3: A typology of ties studied in social network analysis [3].

another's actions in ways that create a particular outcomes"[40]. This concept is called *homophily* and was first introduced by Lazarsfeld and Merton, 1955 [43]. They argue state, that persons with similar characteristics are more likely to be connected and that connected persons are more likely to have common characteristics. *"The connections in a social network affect how people learn, form opinions, and gather news, as well as affecting other less obvious phenomena, such as the spread of disease"*[2].

"Networks are thus a general yet powerful means of representing patterns of connections or interactions between the parts of a system"[2].

2.4.3 The Mathematics of Networks

This subsection describes some basic network measures. In mathematical literature, a network is called graph an consists of vertices and edges. Depending on the field of science, the notation is adapted to nodes and links in computer science or sites and bonds in physics.

The Adjacency Matrix

To present a graph mathematically, we assign each vertex a label from 1 to n. It is important, that each vertex has a unique label, to identify it. The order of the labels is not important. In the next step, we define edges between the vertices i and j as (i, j) and are now able to describe the network in figure 2.4 with n = 6 vertices and the following edges:

- (1,2)
- (1,5)
- (2,3)
- (2,4)
- (3,4)
- (3,5)
- (3,6)

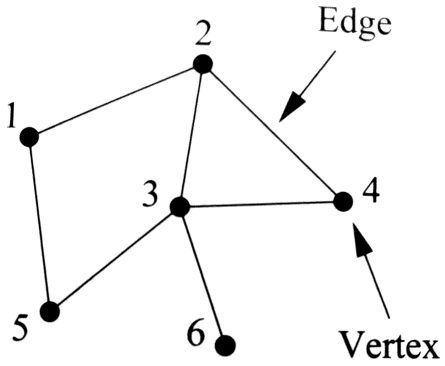


Figure 2.4: A small, undirected network [2]

While this representations is cumbersome for mathematical development, a better way to represent a network is the adjacency matrix. The adjacency matrix contains elements A_{ij} such that

$$A_{ij} = \begin{cases} 1 & if\ there\ is\ an\ edge\ between\ vertices\ i\ and\ j, \\ 0 & otherwise \end{cases}$$

Finally, the adjacency matrix for 2.4 looks like:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{pmatrix}$$

While the graph is connected with undirected edges, the matrix is symmetric i.e. an existing edge between i and j corresponds to an edge between j and i. Additionally, while there are no self-edges (e.g. an edge from i to itself), the diagonal matrix elements are zero. The adjacency matrix can describe networks with self-edges and multiedges too. A multiedge describes multiple connections between two vertices. The value A_{ij} in the matrix equals the number of connections between vertex i and j. A self-edge represents a connection with itself. While an edge has two ends, the value A_{ij} for a self-edge equals 2. An example for a network with self-edges and multiedges looks like figure 2.5: The adjacency matrix describing this network (figure 2.5)[2]:

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 & 3 & 0 \\ 1 & 2 & 2 & 1 & 0 & 0 \\ 0 & 2 & 0 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 0 & 0 \\ 3 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 2 \end{pmatrix}$$

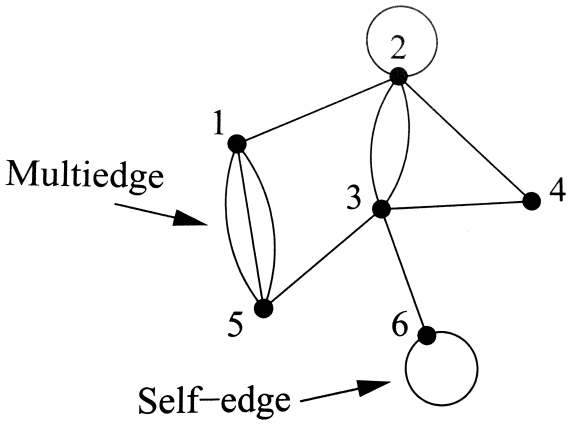


Figure 2.5: A small, undirected network with multiedges and self-edges[2]

Weighted Networks

In addition to simple on/off network connections to describe that an edge between vertices exists or not, it is possible to assign value to edges too. This value can correspond to strength or weight of a link and is usually represented by a real number.

For example, in a social network, the weighted edges could represent the frequency of contact between persons. A simplified adjacency matrix where the connection between person 1 and 2 is twice as frequent as between 1 and 3 looks like [2]:

$$A = \begin{pmatrix} 0 & 2 & 1 \\ 2 & 0 & 0.5 \\ 1 & 0.5 & 0 \end{pmatrix}$$

Directed Networks

"A directed network or directed graph, also called digraph for short, is a network in which each edge has a direction, pointing from one vertex to another"[2]. These so called directed edges are represented as arrows and are mapped to the adjacency matrix with the following logic:

$$A_{ij} = \begin{cases} 1 & if\ there\ is\ an\ edge\ from\ vertex\ j\ to\ i, \\ 0 & otherwise \end{cases}$$

The direction of the edge runs from the second index to the first, although the index is twisted. An example of a directed network looks like figure 2.6:

The corresponding adjacency matrix is [2]:

$$A = \begin{pmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix}$$

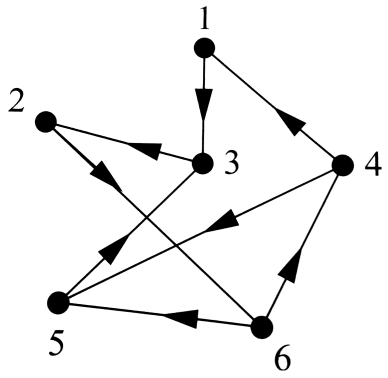


Figure 2.6: A directed network[2]

Directed networks can be used to imitate undirected networks by the use of two edges in each direction where a undirected edge exists. This results in the same symmetric adjacency matrix as known from the undirected networks. Additionally, directed networks can have multiedges and self-edges as well. The only difference is, that self-edges, when these are directed, only count as one in the adjacency matrix because this edge points only to one end. A directed network containing a closed loop of edges, where each of these edges point in the same direction is called a cyclic network (see figure 2.7). Otherwise, it the network is called acyclic.

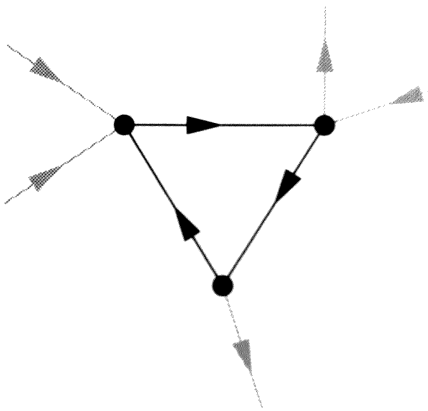


Figure 2.7: A cycle in a directed network[2]

Degree

The number of edges connected to a vertex is called degree of a vertex. The degree of the vertex i is denoted with k_i. The mathematical representation for the degree in an undirected network is described by:

$$k_i = \sum_{j=1}^{n} A_{ij}.$$

While every edge has two ends in an undirected graph, each edge increments the degree of the connected vertices. Therefore, in a network with m edges exists $2m$ ends of edges. While degree counts the number of ends at a vertex and the sum of degrees counts all ends of edges, it is possible to calculate the number of edges based on the degrees in a network:

$$2m = \sum_{i=1}^{n} k_i \rightarrow m = \frac{1}{2}\sum_{i=1}^{n} k_i = \frac{1}{2}\sum_{ij} A_{ij}.$$

The degree is a single number in an undirected network, counting the edges attached to a vertex. In a directed network, the degree is split into *in-degree* and *out-degree*. In-degree counts the number of ingoing edges of a vertex, while the out-degree counts outgoing edges. The adjacency matrix contains $A_{ij} = 1$ when there is a connection from j to i, the different degrees can be written as:

$$k_i^{in} = \sum_{j=1}^{n} A_{ij},\ k_i^{out} = \sum_{i=1}^{n} A_{ij}.$$

Given the number of edges is equal to the number of ingoing edges or equal to the number of outgoing edges, the formula to express this is:

$$m = \sum_{i=1}^{n} k_i^{in} = \sum_{j=1}^{n} k_j^{out} = \sum_{ij} A_{ij}.$$

Centrality

The measure of centrality answers the question, which node is the most important node in the network. Importance in a network depends mostly on the domain. In a computer network, switches and routers are important, in a social network, persons which have a lot of contacts seem to be important. In the simplest case, the centrality is just the degree of a vertex and therefore sometimes called *degree centrality*. An extension to the simple degree centrality is called *eigenvector centrality* where a vertex importance rises, depending on the importance of the connected vertices. *"Instead of awarding vertices just one point for each neighbour, eigenvector centrality gives each vertex a score proportional to the sum of the scores of its neighbours"* [2]. For a social network, this increases one's importance by knowing a lot of people or by knowing people in high places.

Nevertheless, there is a problem with eigenvector centrality, which will be discussed on the following network (see figure 2.8): The problem arises in directed networks, where the adjacency matrix is generally asymmetric. Therefore, in this case, there exists two eigenvectors. The left eigenvector containing ingoing degrees and the right eigenvector is containing outgoing degrees. In most cases, the correct eigenvector to define centrality is the right eigenvector. In social networks or in the World Wide Web (WWW) it is more important to become pointed than pointing on others. Many pages linking to one page makes the one page more important than a page that only presents links to it. So, the right eigenvector defines eigenvector centrality. Based on this decision, vertex A in figure 2.8, which has only outgoing edges will have eigenvector centrality zero. Vertex B has two outgoing and one incoming edge, hence it should

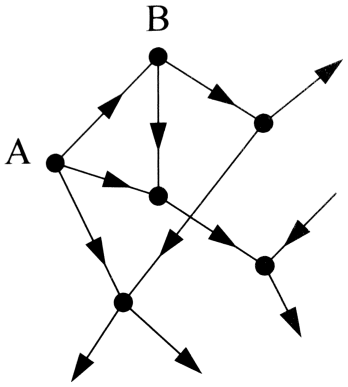


Figure 2.8: A portion of a directed network[2]

have eigenvector centrality one. But while the incoming edge originates from A, the eigenvector centrality of B results in zero too. Finally, a network of lots of vertices could exist, pointing to one node having still an eigenvector centrality of zero.

To overcome this problem, Katz proposed in 1953 to *"give each vertex a small amount of centrality... regardless of its position in the network or the centrality of its neighbours"* [2]. So, every vertex in a network has a non-zero centrality and important vertices are ensured to have a higher centrality measure.

2.5 Visualisation

This section discusses the different graphical representations of the related work projects.

2.5.1 Immerson

This project provides a link to a demo page. The demo randomly selects an existing email account (e.g. Will Hunting, Lex Luthor, Tony Stark, Howard Wolowitz, ...) and starts analysing the stored email correspondence. The top left side of the user interface (UI) (see figure 2.9 contains sliders to modify the layout of the network. The slider at the bottom allows the user to select a timespan for analysis. The right side presents the users name and some statistics in bar-chart representation (email sent, email received and new collaborators).

2.5.2 Prismviz

This project is packaged into a MacOS installer and works on unencrypted iPhone backups to analyse the users SMS[22] history. Prismviz is based on stacked-graphs (see section 2.5.3) and presents plenty of controls to adjust the output (see figure 2.10): The application is available for free (at the moment) but regularly costs 99 cent.

[22]Short Message Service

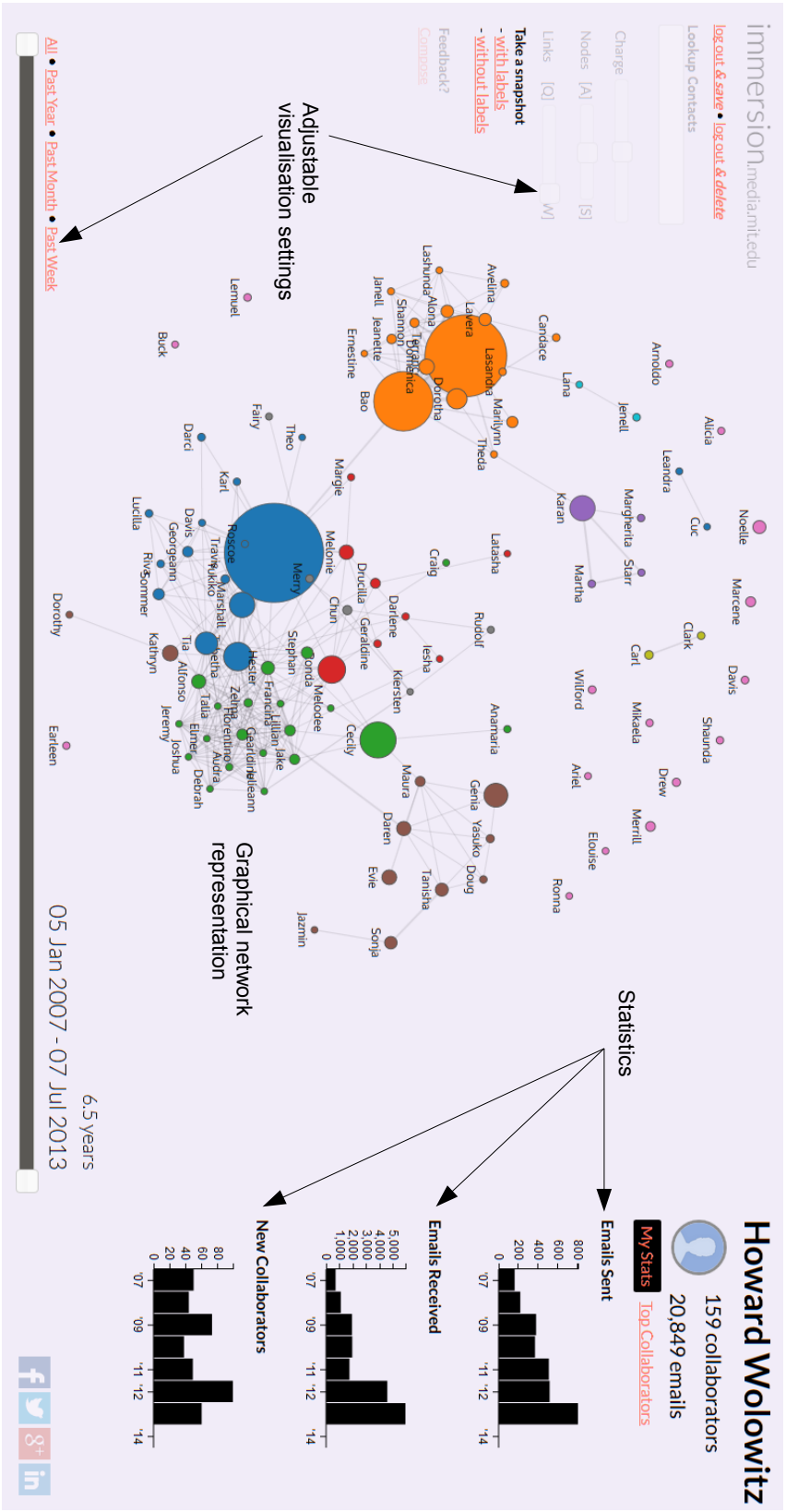


Figure 2.9: Screenshot of Immerson user interface[4].

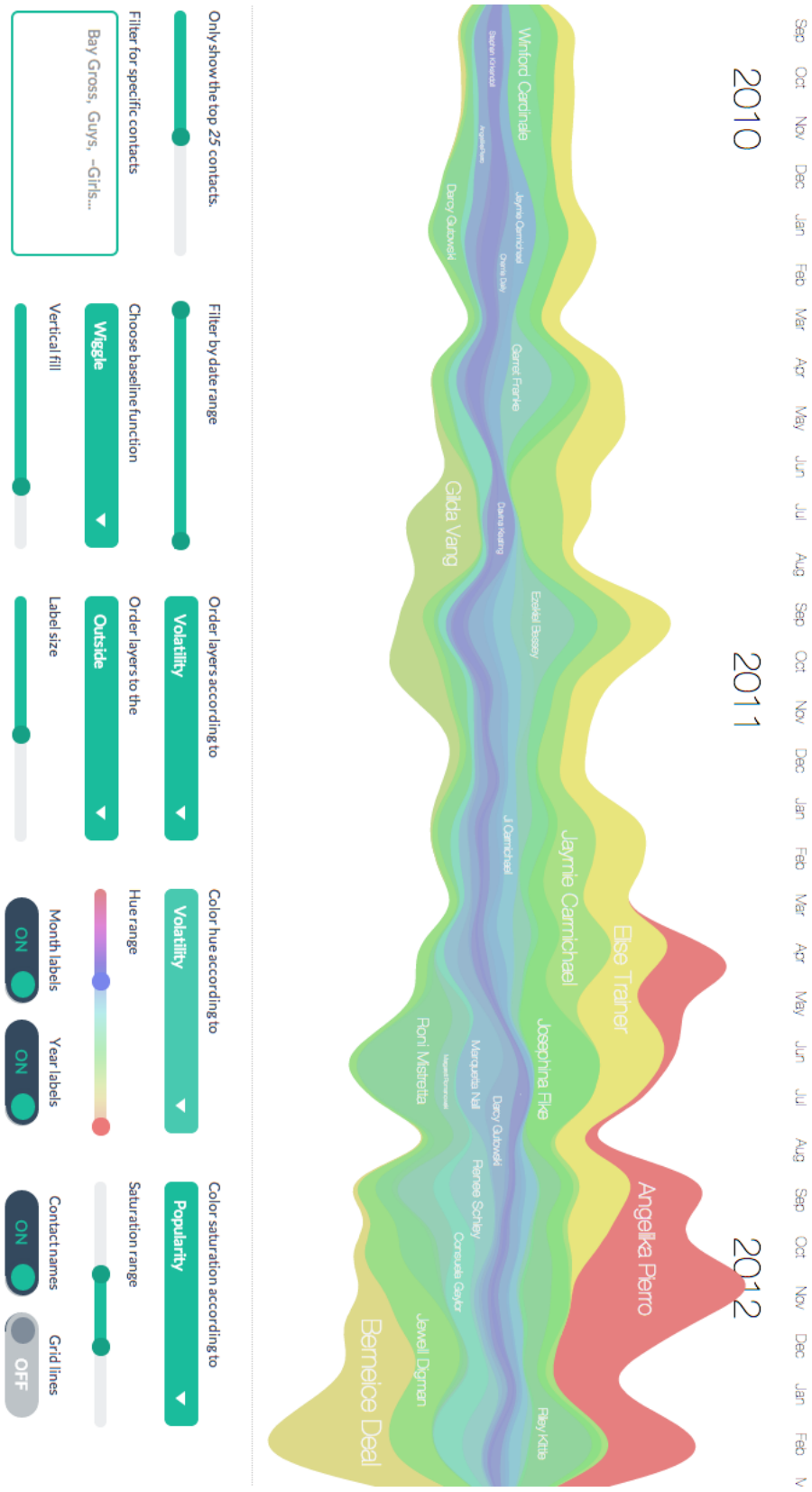


Figure 2.10: Screenshot of Prismviz user interface[5].

2.5.3 Stacked-Graphs

The visualisation of Prismviz is based on the *Stacked-graphs - Geometry & Aesthetics* paper from the authors Lee Byron & Martin Wattenberg [7]. The paper describes the development of this graph and analyses legibility and aesthetics.

Stacked-graphs occurred first, when last.fm[23] decided to visualise trends in personal music listening. Therefore, the author, unhappy with a bar-chart representation of the time series developed an algorithm to create a new form of stacked graphs called stream-graph. *"A stream-graph layout emphasizes legibility of individual layers, arranging the layers in a distinctively organic form"* [7] and had tremendous response in the community. The New York Times kept an eye on that and visualised interactively box office revenue for 7500 movies over a 21-year period (see figure 2.11). Popular

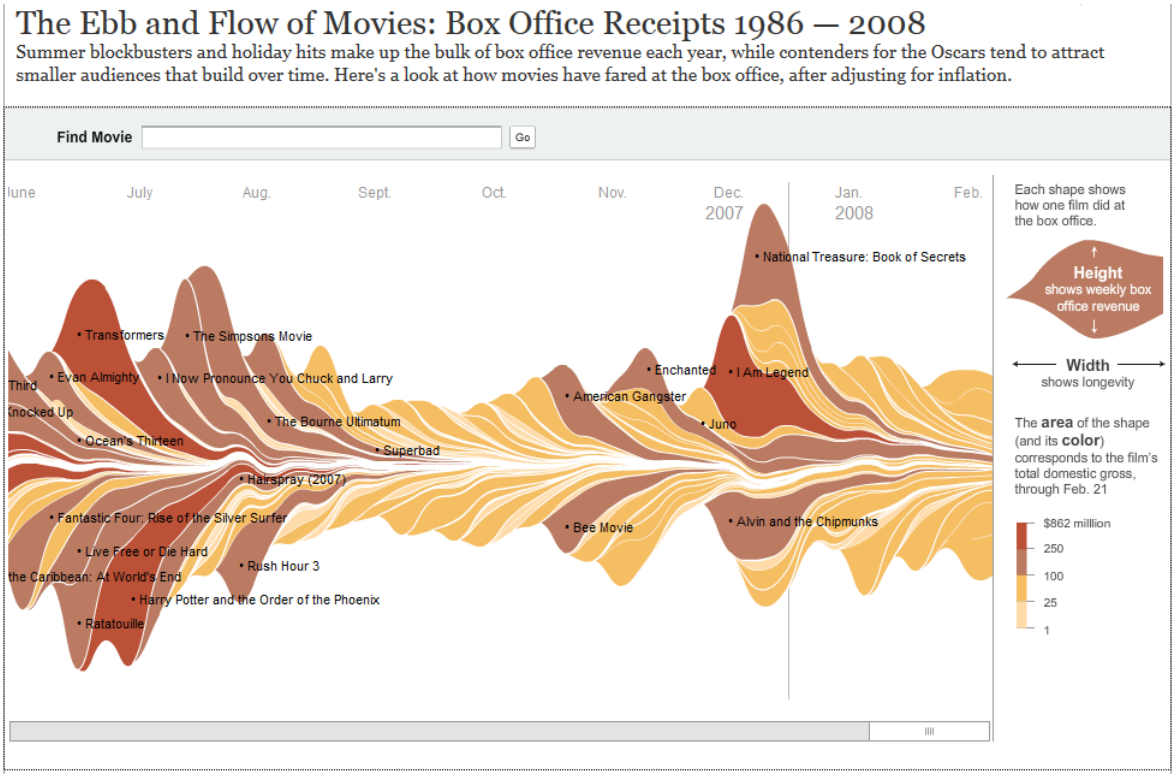


Figure 2.11: New York Times article: The Ebb and Flow of Movies: Box Office Receipts 1986-2008 [6]

reactions showed, that this data representation is helpful for the public as well as for data scientists.

In terms of legibility some issues have to be solved. First, the height of each layer shows its individual value in the time series, whereas the sum of all layers correspond to the total amount of the series at a specific point in time. So, wiggles in one layer influences surrounding, but independent layers. Additionally, different slopes could suggest different relative thickness of layers. Related to this issue is the trade-off between flat and wavy silhouettes. Whereas traditional stacked graphs uses the x-axis

[23] Social music service at `http://www.last.fm`, last seen on 10th July 2014

as bottom line (see figure 2.12), the stream-graph does not, which makes it troublesome to compare the thickness of stream-graphs with different slopes. Finally, the ability to differ between individual layers drops with the number of layers. So, the algorithm has to arrange the different colours in a proper way.

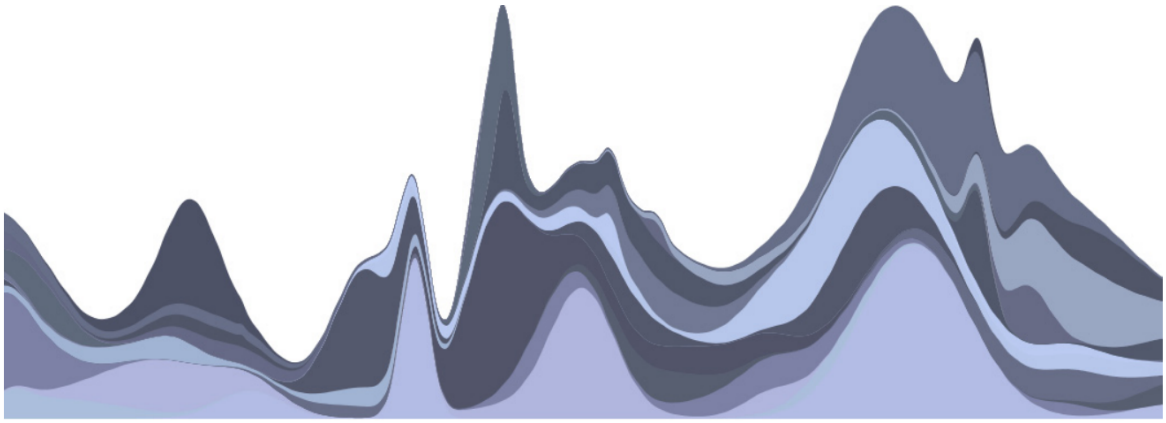


Figure 2.12: A traditional stacked-graph with x-axis as baseline[7]

Related to aesthetics, the representation should attract people to look at the graph and dig into the data. Regarding to the paper, a reason for the positive reaction could be, that this kind of visualisation is not widely spread. As part of aesthetics, the authors conclude their work with discussions about colour selection, placing layer labels and layer ordering. Finally, while the perception between users differ, they ask themselves, what impact varying the discussed parameters for users make.

While this kind of visualisation is still popular for the representation of time series, the paper *Sizing the Horizon: The Effects of Chart Size and Layering on the Graphical Perception of Time Series Visualizations* [44] banishes stacked-graphs from their studies to find the most appropriate technique for presenting time series. These authors argue, that *"though seemingly effective for aggregate patterns, stacked graphs are awkward for comparing individual series"* [44]. Although Byron & Wattenberg suggest *"inside-out"*[7] layer sorting to overcome the different height of layers depending on surrounding layers, the comparison issue remains unsolved.

2.6 Mobile Instant Messaging Applications

This section will introduce the term mobile messaging, present some features of mobile messaging applications and compare them against each other. Finally we will choose one application for analysis purposes.

Instant messaging (IM) describes a form of communication over the internet, where written text can be transferred between subscribers in real time. Current implementations enhanced this pattern by image, audio and video transmission possibilities. While there are a lot of desktop/web applications available for IM (e.g. Skype, Facebook Chat, Google Chat, ...) our thesis concentrates on mobile messaging applications. The TSP

allowed users to use the short message system (SMS) to send a maximum of 160 characters at once to other subscribers. This service is part of the GSM[24] standard [45] and is widely used all over the world. In the year 2012, 646 million[8] messages were sent each month in Austria. Since 2013 the number of text messages is declining[8], while mobile messaging applications become widespread. Combined with the distribution of mobile devices and the availability of mobile broadband connectivity these IM applications replace the short message service more and more (see figure 2.13). Another statistics show, that 56.3% of Austrians use mobile broadband connectivity

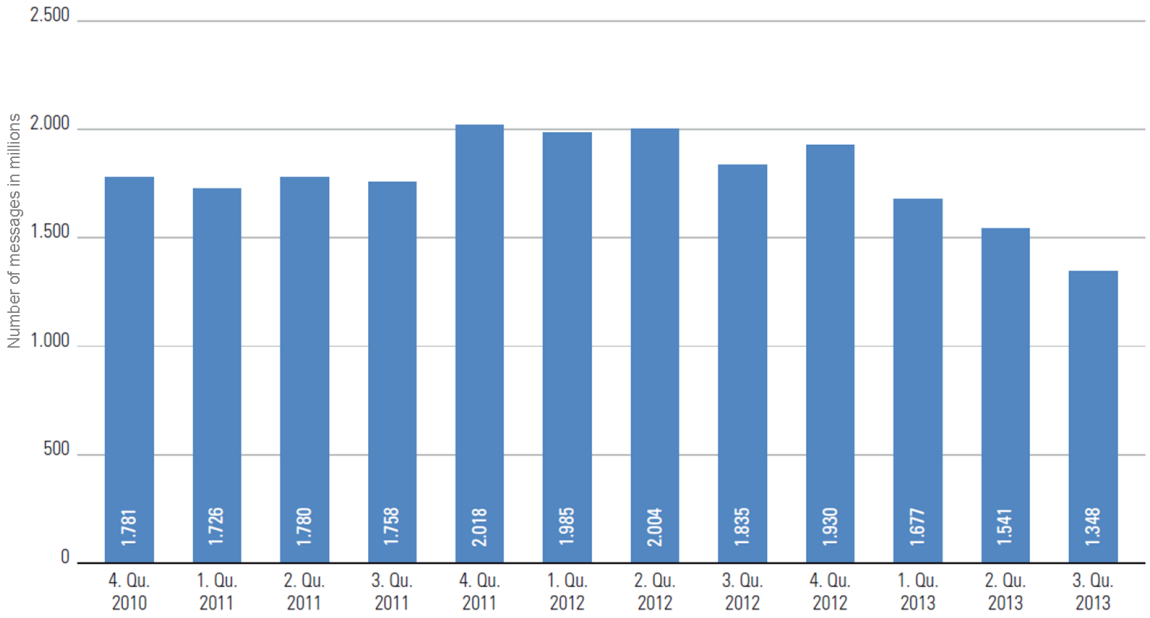


Figure 2.13: Number of sent SMS in Austria from 2010 - 2013[8]

on their smart-phones (84.6% in the age of 16 - 24 years and 75.0 % in the age of 25 - 34 years)[46]. Furthermore, 46.4% uses the internet to participate in social networks (84.8% in the age of 16 - 24 years and 62.6% in the age of 25 - 34) [47].

While there are a lot of messaging applications available, we present a short overview of the most popular applications (see table 2.2). The data of the table was gathered by several search engine requests to identify the most popular messaging applications. Additionally, each available application website was visited and the capabilities as well as the supported platforms extracted (in July 2014).

The column *Installations* refers to the installation count in million according to the application store[25]. The installation count is clustered and can only be determined by the developer of the application in the *Android Developer Console*. *Media capabilities* describes the media recording and transmission capabilities of the application (Text/Image/Audio/Video/Location/Contacts). *Supported platforms* represents

[24]Global System for Communications

[25]Google Play Store is a digital distribution platform for Android applications, music, e-books, movies containing more than 1.3 million mobile applications, `https://play.google.com`, on 10th July 2014

Title	Installations	Media capabilities	Supported Platforms
Whatsapp	500-1000	T/I/A/V/L/C	A/iOS/WP/BB/S
ChatON	100-500	T/I/A/V/L/C	A/iOS/WP/BB/PC
Facebook Messenger	100-500	T/I/A/V/L/C	A/iOS/WP
Google Hangouts	100-500	T/I/A/V/L/C	A/iOS/PC
KakaoTalk	100-500	T/I/A/V/L/C	A/iOS/WP/PC
LINE	100-500	T/I/A/V/L/C	A/iOS/WP/BB/S/PC
Skype	100-500	T/I/-/V/-/C	A/iOS/WP/BB/S/PC
Tango Messenger	100-500	T/I/-/-/-/-	A/iOS/WP/BB
Viber	100-500	T/I/-/-/L	A/iOS/WP/BB/S/PC
WeChat	100-500	T/I/A/-/L/C	A/iOS/WP/BB/S/PC
BBM	50-100	T/I/A/-/L/C	A/iOS/BB
Kik	50-100	T/I/A/V/-/-	A/iOS/WP
Snapchat	50-100	T/I/A/V/L/-	A/iOS/WP
hike Messenger	10-50	T/I/A/V/L/C	A/iOS/WP/BB/S
Telegram	10-50	T/I/-/V/L/C	A/iOS/WP/PC
IM+	1-5	T/I/A/V/L/-	A/iOS/WP/BB
Threema	1-5	T/I/A/V/L/C	A/iOS
ChatSecure	0.1-0.5	T/I/A/-/-/-	A/iOS
TextSecure	0.1-0.5	T/I/-/-/-/-	A

Table 2.2: A list of Mobile Instant Messaging applications

abbreviations for Android/iOS/Windows Phone/Blackberry/Symbian/Personal Computer. The content of the table is sorted according to the number of instanced installed.

2.6.1 Whatsapp

The cross-platform mobile messaging application *Whatsapp* is developed by Whatsapp Inc. since 2008 and is (on the basis of the installation count) the most widespread instant messaging service. According to the Whatsapp homepage[26] this service has 500 million regular users and transmits *"more than 700 million photos and 100 million videos every single day"*[48].

Because of its installation count and its distribution in our social network, we have choosen this application for further investigation.

In February 2014, Facebook acquired Whatsapp for $19 billion[27]. Before the acquisition, Whatsapp faced a security debate because several vulnerabilities where found (e.g. [49], [50]) regarding authentication, message channel encryption, user enumeration and user status retrieval as well as database encryption. Despite these problems, Whatsapp leads the installation count of mobile messaging applications. In spring 2014[28], Whatsapp used their CRYPT5 encryption which could be decrypted by Android appli-

[26] http://www.whatsapp.com/ on 22nd April 2014

[27] http://www.theguardian.com/technology/2014/feb/19/facebook-buys-whatsapp-16bn-deal

[28] no Whatsapp application revision history could be found describing when the cryptographic extensions were introduced. Mail sent to Whatsapp Inc. was not answered

cations (e.g. WhatsApp Tri-Crypt, Whatsapp Crypt-DB Converter). How to decrypt CRYPT5 message history database is referred in section 4.2.7. Successive updates changed the encryption and finally brought the CRYPT7 format, which encrypts the database with a server generated key which is stored on the Android file-system where the user has no access. Section 4.2.7 covers the retrieval of the unencrypted message database to surpass CRYPT7.

Chapter 3

Application Design

This section describes different application designs and the selection of a design for the implementation of the project.

3.1 General Architecture

For simplification issues, the application is divided into three layers:

1. The user interface layer
2. The application layer
3. The data layer

The selection of a layered architectural style has several benefits. First, due to boundaries between layers defined interfaces are required, which lead to lose coupling and exchange ability of components. Second, assignment of functionality to layers are simple. For example, everything that belongs to interaction with the user is encapsulated in the user interface layer. Third, each layer gets information from the layer beyond: *"requests go down the hierarchy whereas the results flow upward"*[9].

The next three subsections briefly describes these layers.

3.1.1 User Interface

The user interface layer will consist of all components required to interact with the user. For this application, the UI has to contain:

- input possibilities for providing of itemised bills[1],
- a list of provided bills, to allow the user to dynamically select and deselect entries used for analysis,
- an output area for the graphical representation of the analysis.

[1] German: Einzelverbindungsnachweise

Additionally, the UI has to handle user events like mouse click or keyboard input and has to provide the user some kind of graphical feedback. Components at this level are typically UI frameworks with defined graphical interfaces and connectors to application layer frameworks.

3.1.2 Application Layer

The application layer, processing or business logic layer computes the user input from the UI, analyses it and uses the data layer to persist it. Subsequent to the analysis, the application layer is responsible for providing the UI with the relevant data required for the visualisation of the results. Components at this level are typically frameworks that connects the UI with the data layer.

3.1.3 Data Layer

The data or persistence layer is responsible for storing new data from the user and providing existing data to the application layer. To store the supplied user data, the business logic transforms it and requests the data layer to store it permanently. Components at this level are typically databases (relational or non-relational) and filesystems.

3.2 Model A - Server Centric Approach

The first model presents a client/server system architecture, where the client is very lightweight (i.e. contains no or little business logic) and therefore called **thin client** according to figure 3.1 (a-c). The client code will run on the client system (e.g. browser) and the application and data layer will run on a separate server system. The client would only consist of the user interface, whereas the server would serve the application and data layer.

Advantages of this architecture for the project are:

- The thin client either contains only logic for presentation issues or little application logic which would reduce dependency and requirements on the client environment
- Server-side storage could aggregate the data of multiple users and expand the social network
- Usage of existing knowledge of Java programming language at the application layer and
- Usage of SQL[2] knowledge at the data layer would speed up implementation process

[2]Structured Query Language, see [51]

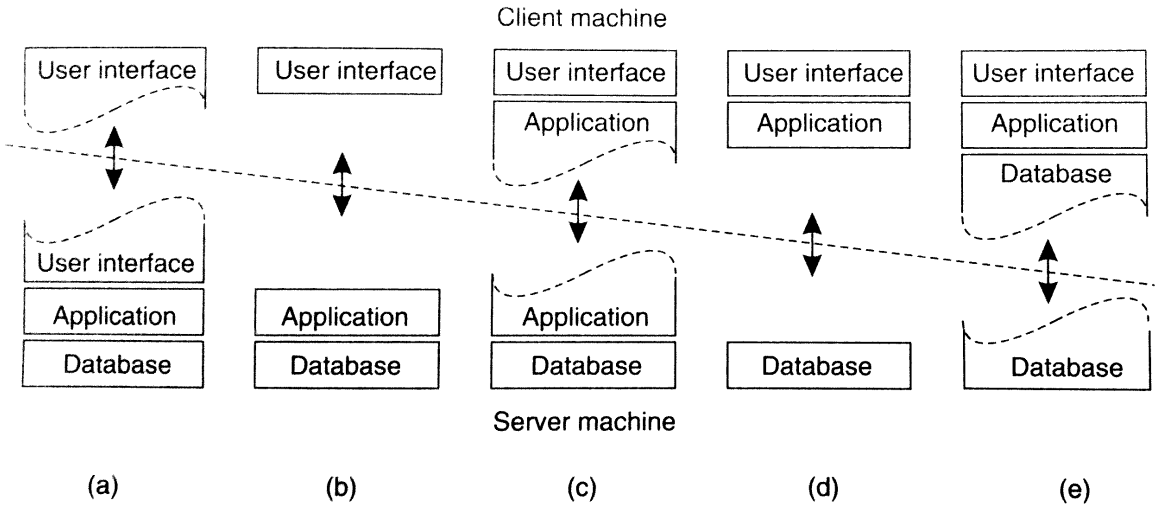


Figure 3.1: Alternative client-server organisations (a)-(e) [9]

So while the advantages lists reduced dependency, sophisticated analysis and accelerated implementation, the disadvantages are:

- Transport and storage of personal data would be challenging
- The server should scale to the number of users
- Low estimated user acceptance to provide personal data

The most problematic item on the disadvantage list would be the transfer to and the storage of personal data in our database. We would have to consider protection of data privacy with all legal consequences. While §9 DSG[3] defines a list of exception (including self publication of personal data), which would us allow to process the users data legally, even so we would be forced to grant the user some rights:

1. Protection against collection[4]
2. Right for disclosure[5]
3. Right for correction or deletion[6]
4. Right to disagree[7]

Apparently, a lot of obligations have to be fulfilled before the project could be publicly used.

[3]German: Datenschutzgesetz, Austrian law for protection of data privacy
[4]German: Erhebungsschutz
[5]German: §26 DSG Auskunftsrecht
[6]German: §27 DSG Recht auf Richtigstellung oder Löschung
[7]German: §28 DSG Widerspruchsrecht

3.3 Model B - Client Centric Approach

The second model introduces a client only implementation where all layers according to 3.1 (d-e) would run on the client system. This architecture is called **thick** or **fat client**. All parts required to run the application would be initially loaded from the web server and then started on the client system of the connected user. After the initialisation, this implementation requires no further connection between the client and the web server.

Advantages of this architecture for the project are:

- Communication between client and server system only at initialisation
- No scaling problem → burden of analysis moved from server to client system
- No legal obligations for transport and storage of personal data
- Personal data do not cross the wire → higher user acceptance

Beside of the advantages, some disadvantages remain:

- High dependency on client environment
- Advanced analysis not possible
- Knowledge of required technologies is low

Therefore, the most problematic disadvantages of Model A can be overcome at the cost of implementing in a new and relatively unknown environment.

3.4 Decision

The decision between the models was not easy. On the one hand well known environment but legal problems with Model A. Unknown environment on the other hand. While the legal problems are not relevant in a private environment, it would definitely limit acceptance of the project outside the development team. Furthermore, development in recent years created sophisticated client-side technologies, that should leverage our implementation. Finally, to create benefit for and get maximum acceptance from other users, we favour Model B.

Chapter 4

Implementation

After discussion and selection of the preferred application design, the project has to be implemented. This section describes the selected components and the implementation process step by step and will go into detail on relevant parts.

4.1 Component selection

This section describes the selected components for our implementation.

After choosing *Model B* (see section 3.3) as our preferred architecture, the required components have to be listed and concrete implementations have to be selected. According to *Model B*, all layers of the fat client (see figure 3.1 (e)) runs on the client system. Based on the experiences with J2EE[1] and J2SE[2] the implementation as an Java applet would be reasonable. Nevertheless, recurring security problems of Java in association with web browsers have lowered user acceptance. Regular security updates of Oracles JVM[3] increases the possibility, that our implementation could become unsupported with one of the updates. Additionally, many browser vendors disable the required Java plugin, so that Java applets are not able to execute. Finally, mobile platforms support J2ME[4] only. Therefore, the applet would not be able to execute in this environment.

Under these circumstances, dropping usage statistics of Java applets[5] and the rise in usage of JavaScript in open source projects (see table 4.1) and websites[6], JavaScript will have a fundamental role in the implementation. JavaScript not only runs on desktop browsers, it is also available on mobile devices. So our application can reach as much clients as possible.

[1] Java2 Enterprise Edition

[2] Java2 Standard Edition

[3] Java Virtual Machine, the runtime environment of Java programs

[4] Java2 Micro Edition

[5] see analysis of websites using Java applets at `http://trends.builtwith.com/docinfo/Applet/`, last seen on 10th July 2014

[6] see analysis of websites using JavaScript libraries at `http://trends.builtwith.com/docinfo/Javascript/`, last seen on 10th July 2014

Rank	Language	# Repositories created
1	JavaScript	264131
2	Ruby	218812
3	Java	157618
4	PHP	114384
5	Python	95002
6	C++	78327
7	C	67706
8	Objective-C	36344
9	C#	32170
10	Shell	28561

Table 4.1: GitHub.com statistics showing top 10 languages in 2013[13]

4.1.1 HTML5

Hyper**T**ext **M**arkup **L**anguage revision 5 (HTML5) is a core technology of the World Wide Web (WWW) and is used to present content on the internet in a structured way. The 5^{th} revision subsumes not only its predecessor HTML4 but also XHTML[7] 1 and DOM[8]. XHTML 1 describes a more restrictive subset of SGML[9] than HTML and XHTML documents have to be well-formed in comparison to HTML documents processed by lenient parsers. DOM describes a set of rules how to represent and interact with objects in HTML documents and is used to dynamically manipulate the content of these documents. The current release plan of the Word Wide Web Consortium[10] (W3C) schedules its recommendation for HTML5 at the end of 2014. Nevertheless, the main browser vendors have adopted to HTML5 and the number of websites using it rose in recent years[11].

Additionally to using state-of-the-art technologies for the implementation, HTML5 is required because it contains some APIs[12] that are crucial for the project (see figure 4.1):

To store the users data on the client-side, our project will use IndexedDB (see section 4.1.4). For visualising the results of the analysis, D3.js (see section 4.1.5) uses SVG[13] to render the charts.

[7] eXtensible HyperText Markup Language

[8] Document Object Model

[9] Standard Generalized Markup Language, a flexible markup language framework

[10] The W3C is the main international standards organisation for the WWW founded by Tim Berners-Lee

[11] see analysis of websites using HTML5 at `http://trends.builtwith.com/docinfo/HTML5-DocType`, last seen on 10th July 2014

[12] Application Programming Interface that describes how to interact with a software component

[13] Scalable Vector Graphics

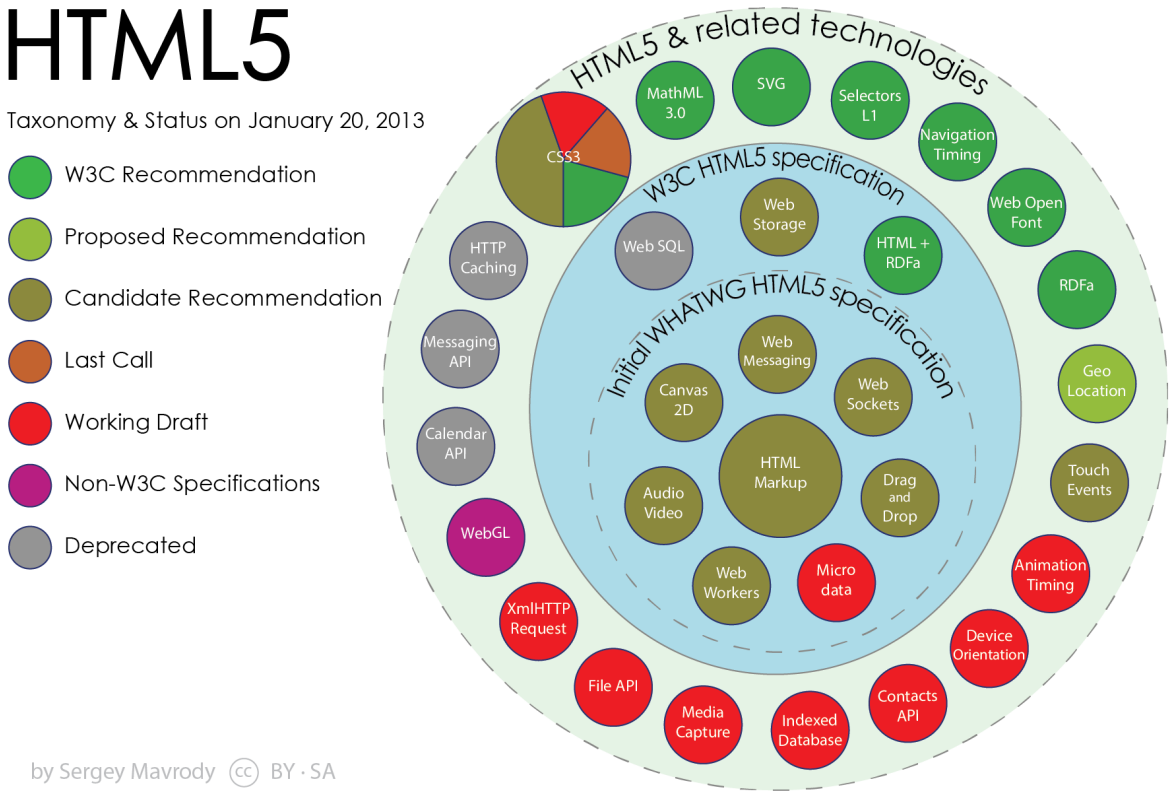


Figure 4.1: HTML5 APIs overview [10]

4.1.2 AngularJS - Model-View-Controller Framework

AngularJS [52] is a Model-View-Controller (MVC) framework written in JavaScript, maintained by Google and distributed under the MIT licence. It allows developers to define new HTML elements and *"is extraordinarily expressive, readable and quick to develop"*[52]. The decision to use a MVC framework was driven by the assumption, that implementing the project will cause a lot of written code and therefore needs some framework in place to

- structure the code,
- separate view from model and
- provide standard functionality.

Analysis of existing JavaScript MVC frameworks has lead to BackboneJS and AngularJS. Both are found in the Google Search results and are compared to each other frequently. While both frameworks are completely unknown to the development team, some usage statistics were analysed[14]. A comparison between both frameworks presents figure 4.2.

[14]see analysis of websites using AngularJS at `http://trends.builtwith.com/javascript/Angular-JS`, last seen on 10th July 2014, with BackboneJS at `http://trends.builtwith.com/javascript/Backbone.js`, last seen on 10th July 2014

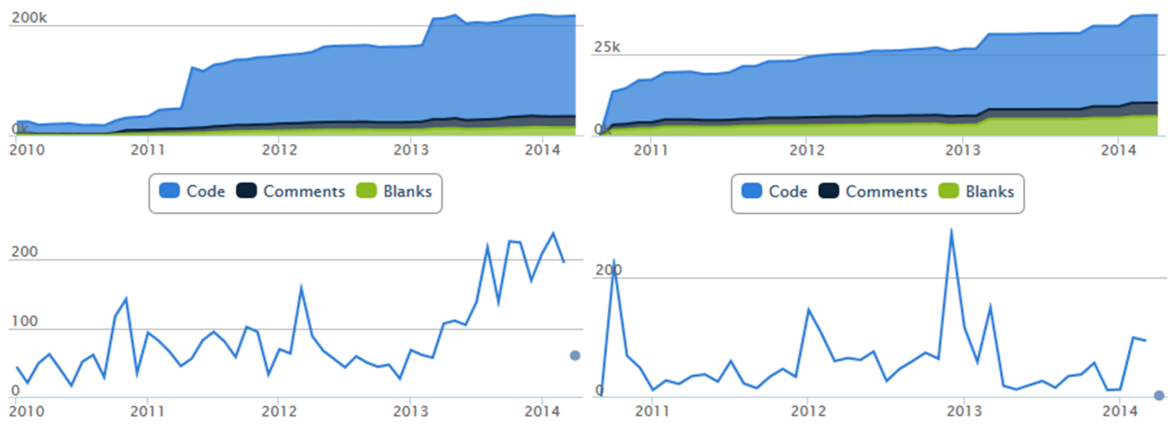


Figure 4.2: Comparison between AngularJS (left) and BackboneJS (right) in terms of lines of code (top) and commits per month (bottom)[11].

Finally, the decision in favour of AngularJS was reached, while the AngularJS project

- has good documentation,
- is backed by Google,
- has a tremendous developer community and
- supports simple data-binding between model and view.

AngularJS uses the following main components to achieve separation of concerns:

1. Module
2. Controller
3. Service
4. Directive.

AngularJS uses *modules* as a container for all other parts of an application (e.g. controllers, services, directives, ...). While most common applications have a main method, bootstrapping and wiring the components, AngularJS claims to have some advantages with the modules approach. This approach should increase the reuse of code, allows programmers to coordinate load order of program parts and improves testability. Tests are influenced because for each test only the required modules need to be loaded, which reduces test duration.
A *scope* in AngularJS is an object that refers to the application model as an execution context for expressions. Each website has a root scope which can contain multiple other scopes. To create a new child scope, a *controller* is used to

- set up the initial state of a scope and

- add behaviour to a scope.

The initial state is represented by variables, whereas behaviour is achieved by the use of functions. These variables and functions can only be used inside the scope. So, different scopes can help so separate specific areas of a website. For communication between scopes, programmers can use services.
A *service* in AngularJS is an singleton object, that is lazily instantiated and wired together by dependency injection (DI). A singleton is an object, that can only be once instantiated. Each subsequent instantiation will return the same object. An lazy instantiation only loads an object when it is needed and not on program start. Finally, DI is a software design pattern, describing how an object gains its dependencies. AngularJS creates services with the *factory* keyword which creates the instance at the time it is first called.
The last component called *directives* acts as a placeholder for more specific behaviour in HTML. During the bootstrap of the application, the AngularJS HTML compiler attaches behaviour to or replaces the directive element in the DOM. In this way, it is possible to define new HTML tags, with unique attributes (variables in the AngularJS directive description) and behaviour (corresponding to directive functions).

4.1.3 FileReader API

The FileReader API[15] is part of HTML5 (see figure 4.1) and allows the browser to read local files provided by the user. It is supported by current versions of major browsers[16] and is required in this project to read the itemised bills from the users and provide their content to the IndexedDB (see section 4.1.4).

To read text files (like CSV files), FileReader provides a function called *FileReader.readAsText()*. For binary files (like Whatsapp database file) the function *FileReader.readAsBinaryString()* can be used. FileReader itself asynchronously reads the content of a file after calling one of these functions. When the function succeeds, the supplied *onLoad*-function (callback) is executed with the content as a parameter.

4.1.4 IndexedDB - Client-side Storage

In the specification of the HTML5 standard (see figure 4.1), client-side storage has been introduced. Model B (see section 3.3) requires client-side storage to buffer the provided data for the analysis process. So, IndexedDB is a crucial component to put Model B into practice. IndexedDB is a NoSQL[17] database for storing and retrieving objects. For the analysis of multiple files supplied by the user, the application has to buffer it. When loading data from the user, the file is parsed and the relevant information is stored in the client-side IndexedDB. Finally, the visualisation can access the data through application logic and draw the corresponding charts. The content of the database is lost on page reload. The browser Google Chrome is able to inspect

[15] http://www.w3.org/TR/FileAPI/
[16] http://caniuse.com/filereader
[17] Not-Only-SQL describes a mechanism to work with data other than in tuples like relational databases

the IndexedDB with the help of its developer console which lists the stored key-value pairs.

4.1.5 D3.js Visualisation Library

To visualise the results of the analysis of the user input, a visualisation component is required. Starting point was Gephi [53], which is an open-source, interactive visualisation and exploration platform. Nevertheless, Gephi is a desktop application and not able to run in a browser. So it cannot be used in this project. Finally, D3.js [54] has been found, which commits itself to interactive data visualisation for the web. D3.js is the abbreviation of Data-Driven Documents and is primarily written by Mike Bostock. The library is released under a BSD licence and is the successor of Protovis[18] which was developed by Jeff Heer and Mike Bostock at Stanford's Vis Group in 2009. Finally, in 2011 D3.js was announced[55] to support three objectives:

- Compatibility,
- Debugging and
- Performance.

Based on HTML5 and its improved support for SVG, D3.js loads faster than equivalent Flash visualisations[55], which is, additional to the BSD licence a reason to use this library in this project. While D3.js is the successor of Protovis, D3.js has a lot of attention from the community and (like AngularJS) a remarkable documentation. The potential of D3.js, the documentation[19] and the vast extent of examples available on the internet were crucial for the selection of D3.js as the project's visualisation library.

4.2 Implementation

This section describes in detail the implementation process. The most relevant parts of the implementation are described in more detail.

4.2.1 Preparation

In the first step, to get in touch with AngularJS, download the library and run through the tutorial[20]. It is easy to follow because it is well written. The second step consists of the composition of the final site layout. Finally, by using the described components (see section 4.1) the website has been implemented. The project layout looks like (see listing 4.1):

[18] http://mbostock.github.com/protovis/

[19] `https://github.com/mbostock/d3/wiki`, last seen on 10th July 2014

[20] `https://docs.angularjs.org/tutorial`, last seen on 10th July 2014

```
metacommunicationjs
+ css
  - app.css
  - bootstrap.min.css
+ js
  + vendor
    - angular.min.js
    - angular.min.js.map
    - d3.min.js
    - pdf.js
    - pdf.worker.js
    - sql.js
    - ui-bootstrap-tpls-0.11.0.min.js
  - app.js
  - controllers.js
  - directives.js
  - reusableBarChart.js
  - reusableForceGraph.js
  - reusableStreamGraph.js
  - services.js
- index.html
```

Listing 4.1: Project layout on file-system

The application is split into the following packages (see figure 4.3):

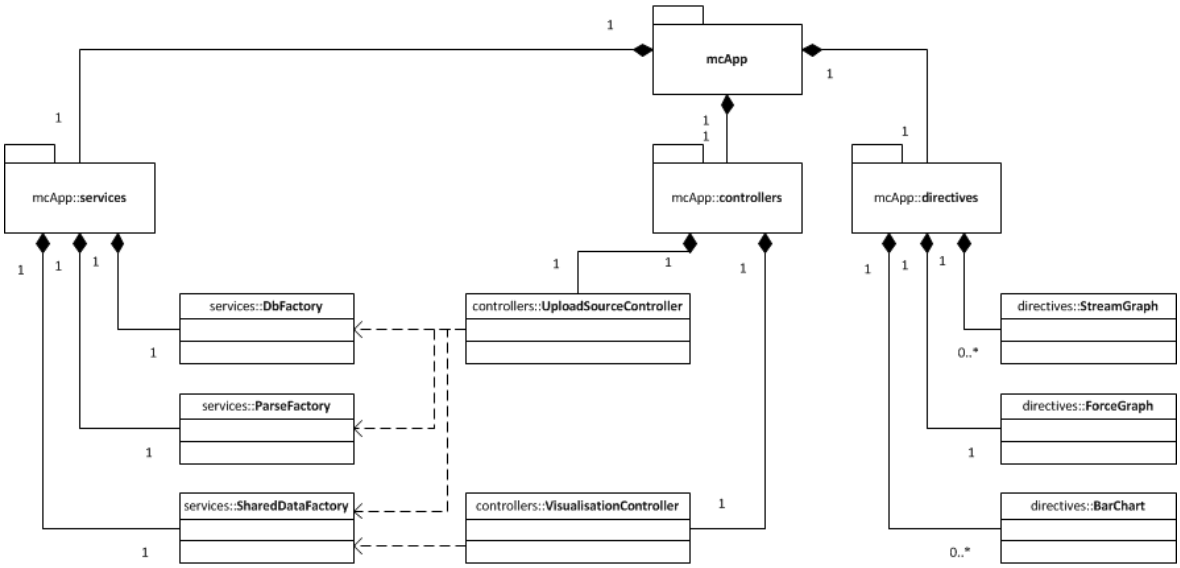


Figure 4.3: Application model - An overview

The connection between HTML5 code and the Javascript implementation is made with the top level module name: mcApp (MetaCommunicationApp). This module contains services, controllers and directives (see section 4.1.2). This separation improves code structure and reuse of the code. The *controllers* puts the MVC paradigm into work. It contains the model and the controller of the MVC pattern and connects it with the HTML5 code, the view. *Controllers* can not everything, therefore it uses *services* (business logic, model related). The *directives* are part of the view by adding additional behaviour to HTML5.

4.2.2 Website Layout

To provide the users a simple and self-explanatory interface (see 4.4) for uploading their data, the components have to be clearly arranged. To get modern looking controls we are using Bootstrap[21]. This component consists of a JavaScript library and a Cascading Style Sheet (CSS) to modify the visual representation and provides our website a modern look and feel. Finally, we use a grid layout, which should perform on desktop as well as on mobile devices with reduced screen sizes. The code for the basic layout is shown in listing 4.2.

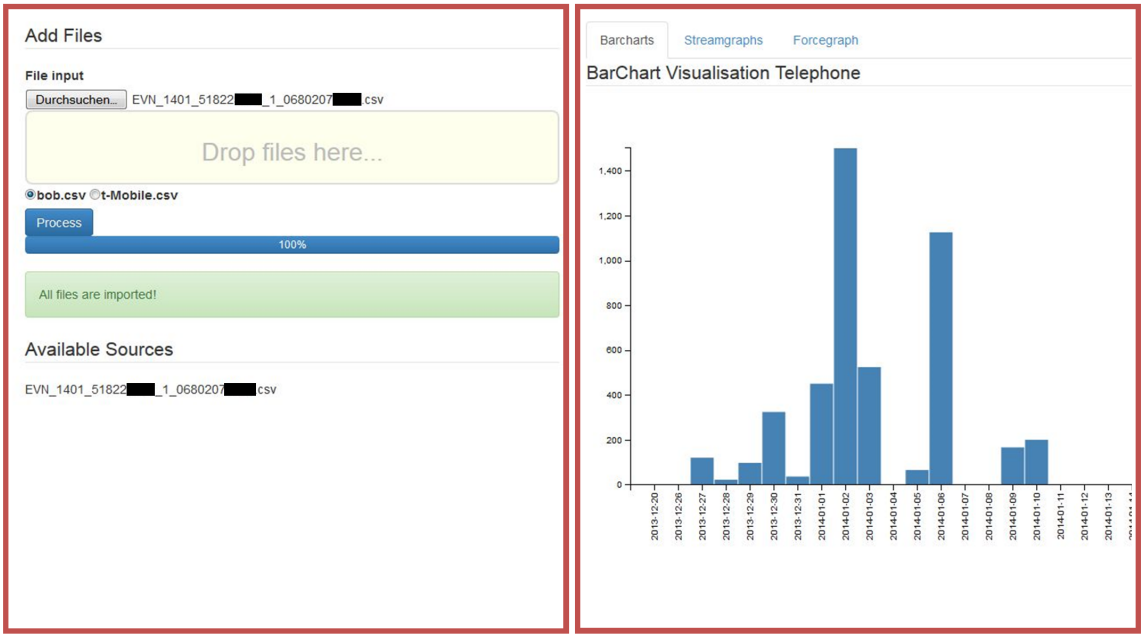


Figure 4.4: Website layout with file input on the left side and visualisation on the right side

```
1  <!DOCTYPE html>
2  <html ng-app="mcApp">
3  <head>
4    <meta charset="utf-8">
5    <title>Metacommunication</title>
6    <link rel="stylesheet" href="css/bootstrap.min.css">
7    <link rel="stylesheet" href="css/bootstrap-theme.min.css">
8    <link rel="stylesheet" href="css/app.css">
9  </head>
10 <body>
11   <div class="container-fluid">
12     <div class="row">
13       <div class="col-md-4" ng-controller="UploadSourceController">
14         <fieldset>
15           <legend>Add Files</legend>
16         </fieldset>
```

[21] http://getbootstrap.com/, last seen on 10th July 2014; *Bootstrap makes front-end web development faster and easier. It's made for folks of all skill levels, devices of all shapes, and projects of all sizes.*

```
17
18          <fieldset>
19            <legend>Available Sources</legend>
20          </fieldset>
21        </div>
22        <div ng-controller="VisualisationController">
23        <tabset class="col-md-8">
24          <tab heading="Barcharts">
25            <bar-chart service="TEL" data="data"></bar-chart>
26            <bar-chart service="SMS" data="data"></bar-chart>
27            <bar-chart service="DATA" data="data"></bar-chart>
28          </tab>
29
30          <tab heading="Streamgraphs">
31            <stream-chart service="TEL" data="data"></stream-chart>
32            <stream-chart service="SMS" data="data"></stream-chart>
33            <stream-chart service="DATA" data="data"></stream-chart>
34          </tab>
35
36          <tab heading="Forcegraph">
37            <force-graph data="data"></force-graph>
38          </tab>
39        </tabset>
40        </div>
41      </div>
42
43      <footer>
44      </footer>
45    </div>
46
47    <script src="js/vendor/d3.js"></script>
48    <script src="js/vendor/angular.min.js"></script>
49    <script src="js/vendor/ui-bootstrap-tpls-0.11.0.min.js"></script>
50    <script src="js/reusableBarChart.js"></script>
51    <script src="js/reusableStreamChart.js"></script>
52    <script src="js/reusableForceGraph.js"></script>
53    <script src="js/app.js"></script>
54    <script src="js/services.js"></script>
55    <script src="js/directives.js"></script>
56  </body>
57  </html>
```

Listing 4.2: HTML5 code for the layout

The listing 4.2 contains the HTML5 Doctype definition in line 1. While different versions of the HTML Standard exists, browsers need to now which standard should be used to render the website. The Doctype definition in the first line switches the browser to HTML5 standard mode. The <*head*> section contains the title of the document, the characterset definition and some stylesheet links. The title is shown in the title bar of the browser after accessing the website while the characterset definition is required to set the character encoding of the website. Stylesheets allows designers to modify the look and feel of a website with a special markup language called Cascading Style Sheets (CSS). In this case we provide two files for Bootstrap and one file for special formatting of the visualisation graphs (line 8).

The *body* section in the HTML code includes a *container-fluid* definition which is part of Bootstrap's grid system[22] and itself contains rows and columns.

AngularJS uses the attributes *ngapp="mcApp"* in line 2 to link to the corresponding AngularJS module[23] definition. Because a website can contain more than one AngularJS application, this attribute is required. Further separation is based on the usage of AngularJS controllers[24]. The HTML attribute *ngcontroller="UploadSourceController"* in line 13 and *ngcontroller="VisualisationController"* in line 22 of the *div* elements is used to bind these sections to AngularJS controllers which limit the scope of methods and properties. The tag *tabset* is Bootstrap specific and provides a simple tab switching layout for the visualisation. The visualisation itself is wrapped into reusable AngularJS directives[25]. These tag contains a data-property and optionally a service-property. The bar-charts as well the stream-graphs are separated by service. The force-graph integrates all services.

The *body* tag closes with the imports of different JavaScript libraries required for the MVC Framework (AngularJS), visualisation (D3.js), Bootstrap and the implemented charts and graphs.

4.2.3 AngularJS

AngularJS is triggered by the *ng-app* attribute of the *html* tag and connects the HTML5 with the JavaScript part.

```
1 var app = angular.module('mcApp',
2     ['mcApp.services', 'mcApp.directives', 'ui.bootstrap']);
```

Listing 4.3: AngularJS module definition and inheritance

AngularJS uses its own namespace and is addressed by the keyword *angular*. The value of the *html ng-app* attribute is the module name (see listing 4.3, line 1). In the second line, the services and directives are referenced as well as Bootstrap specific code.

The AngularJS module contains two controllers (see listing 4.4).

```
1 app.controller('UploadSourceController',
2     function($scope, ParseFactory, DbFactory, SharedDataFactory) {
3 });
4
5 app.controller('VisualisationController',
6     function($scope, SharedDataFactory) {
7 });
```

Listing 4.4: AngularJS controller definition

A Controller is used for partitioning the source code and provide a separated scope for a group of actions. Our application uses only two controllers:

- to handle the file upload and data insertion into IndexedDB

[22] `http://getbootstrap.com/css/#grid`, last seen on 10th July 2014
[23] `https://docs.angularjs.org/guide/module`, last seen on 10th July 2014
[24] `https://docs.angularjs.org/guide/controller`, last seen on 10th July 2014
[25] `https://docs.angularjs.org/guide/directive`, last seen on 10th July 2014

- to inform the visualisation components that new data is available.

The most important part of the *UploadSourceController* is the function *processFiles* (see listing 4.5):

```
1  $scope.processFiles = function()
2    DbFactory.openDatabase().then(function() {
3      for (var i in $scope.files) {
4
5        ParseFactory.parseFile($scope.files[i], $scope.inputFormat).then
             (
6          function (resultParse) { // parsing successful
7
8            DbFactory.insertData(resultParse).then(
9              function (resultInsert) { // insert successfull
10
11               DbFactory.selectData().then(
12                 function (resultSelect) { // selection of whole
                       database successfull
13
14                   SharedDataFactory.setDatabaseValues(resultSelect);
                       // set data and trigger visualisation
15
16                   var sources = d3.set(SharedDataFactory.
                       getAvailableSources());
17                   resultSelect.forEach(function(d) { sources.add(d.
                       source); });
18                   SharedDataFactory.setAvailableSources(sources.values
                       ()); // set source list
19
20                 }, function (resultSelect) {}); // selection failed
21             }, function (resultInsert) {}); // insert failed
22         }, function (resultParse) {}); // parsing failed
23     } // end for loop
24   }); // open Database
25 }; // end function
```

Listing 4.5: Workflow of the import part

This part opens the IndexedDB for write access. Then it iterates over all supplied files and parses the content. Depending on the radio-buttons in the UI, the parser chooses the parsing format and retrieves the file contents as an array of objects. Each object contains one row of a CSV file. If parsing is successful, the array of objects is inserted into the IndexedDB. After inserting the last file in the list, all rows are selected from the database and provided to the *VisualisationController*, which triggers the visualisation components. Finally, the list of available sources is set.

4.2.4 FileReader API

The usage of the FileReader API is implemented with a *promise*. A promise is a JavaScript asynchronous programming pattern, that helps to simplify the traditional callback problem. A callback is an other programming pattern, that is used to achieve asynchronous programming in JavaScript. Instead of waiting for a function to return, a

callback function is passed over to the asynchronous function and is finally called, when the action of the asynchronous function is finished. So the asynchronous function call becomes non-blocking. The resulting problem is, that the source code looks scrambled when multiple asynchronous functions are chained. Therefore the promise abstraction pattern was invented (see listing 4.6, lines 3, 31, 35 and 44). A promise is an object that represents a task which will finish in the future (or not).

```
1  app.factory('ParseFactory', function($q) {
2    parseFile : function(file, inputFormat) {
3      var deferred = $q.defer(); // promise
4      var fileReader = new FileReader();
5
6      fileReader.onload = function(e) {
7        var content = {};
8        if (inputFormat !== undefined) {
9          if (inputFormat != "sqlite") { // CSV --> text
10           var csv = fileReader.result;
11           // \r\n == windows style; \n == unix style line breaks
12           var allTextLines = csv.split(/\r\n|\n/); // extract lines
13           var lines = [];
14           for (var i = 0; i < allTextLines.length; i++) {
15             var data = allTextLines[i].split(';'); // extract columns
16             var tarr = [];
17             for (var j = 0; j < data.length; j++) {
18               tarr.push(data[j]);
19             }
20             lines.push(tarr);
21           }
22           if (inputFormat == "csvBOB") {
23             content = global.parseBobFile(file.name, lines);
24           } else if (inputFormat == "csvTMOB") {
25             content = global.parseTMobileFile(file.name, lines);
26           }
27         } else { // sqlite --> binary
28           content = global.parseSQLiteDB(file.name, fileReader.result)
                 ;
29         } // end sqlite
30       } // end if !undefined
31       deferred.resolve(content);
32     }; // end fileReader.onload
33
34     fileReader.onerror = function(e) {
35       deferred.reject();
36     } // end fileReader.onerror
37     if (inputFormat !== undefined) {
38       if (inputFormat != "sqlite") {
39         fileReader.readAsText(file);
40       } else {
41         fileReader.readAsBinaryString(file);
42       }
43     }
44     return deferred.promise;
45   } // end parsefile
46 } // end ParseFactory
```

Listing 4.6: Usage of the FileReader API

The initialisation of the promise is shown in line 3 of listing 4.6. It is implemented in AngularJS and obtained from $q and uses the Deferred and Promise API. After asking the file reader to process an input file (lines 39 and 41) the promise is retrieved from the deferred reference and returned to the calling function. In case the file reading process terminates successfully, the deferred is resolved to the content of the file (line 31). Otherwise, the deferred is rejected (line 35).

The calling function uses the promise to control the program flow (see listing 4.7):

```
1 ParseFactory.parseFile().then(
2   function (result) {
3     // success
4   },
5   function (result) {
6     // error
7   });
```

Listing 4.7: Usage of promise

This construct simplifies the source code and improves readability of the main program. Additionally to the usage of promises, the *parseFile* function differs between itemised bills and Whatsapp SQLite database files. This behaviour is triggered by the radio buttons in the UI (see figure 4.4). Processing of files can only start, if one radio button is selected and at minimum one file is supplied.

4.2.5 IndexedDB

Like the FileReader API, the usage of IndexedDB is encapsulated in an AngularJS service. The service exports the following functions:

- openDatabase
- selectData
- insertData
- deleteData
- resetDatabase

The service is defined within a factory function *DbFactory* (line 1 in listing 4.8) and retrieves the IndexedDB handler from the browser. The corresponding function call differs between browser manufacturers. The default function is *indexedDB* (line 2), for Mozilla based browsers *mozIndexedDB*, for Google based browsers *webkitIndexedDB* and for Microsoft based browsers the function is named *msIndexedDB* (see listing 4.8, line 2).

```
1 app.factory('DbFactory', function($window, $q) {
2   var indexedDB = $window.indexedDB || $window.mozIndexedDB || $window
      .webkitIndexedDB || $window.msIndexedDB;
3   var db = null;
4   // declare error function
5   indexedDB.onerror = function(e) {
```

```
6       console.log("indexedDB.onerror=", e);
7     }
8     var openDatabase = function(){
9       var deferred = $q.defer(); // promise
10      var version = 1;
11      var request = indexedDB.open("metadataDB", version);
12      // ONUPGRADENEEDED
13      request.onupgradeneeded = function(e) {
14        db = e.target.result;
15        e.target.transaction.onerror = indexedDB.onerror;
16        // delete existing store
17        if(db.objectStoreNames.contains("metadata")) {
18          db.deleteObjectStore("metadata");
19        }
20        // create new store
21        var store = db.createObjectStore("metadata",
22          {keyPath: "id", autoIncrement: true});
23        // create indizies
24        store.createIndex("from", "from", { unique:false });
25        store.createIndex("to", "to", { unique:false });
26        // create more indizies
27      };
28      // ONSUCCESS
29      request.onsuccess = function(e) {
30        db = e.target.result;
31        deferred.resolve();
32      };
33      // ONERROR
34      request.onerror = function(e){
35        console.log("openDatabase.onerror=", e);
36        deferred.reject();
37      };
38      return deferred.promise;
39    }; // end openDatabase
```

Listing 4.8: Usage of the IndexedDB

The function *openDatabase* (line 8) requests to open a database called *metadataDB* in a specific version. If a database does not exist or the version differs, the *onupgradeneeded* function is called. While our implementation deletes the database on reload, it should never exist when line 11 is executed. To be sure, line 17 checks for an object store called *metadata* and deletes it, before our application creates a new one to store the user data. The object store is created in line 21 and contains an auto increment field named *keypath*. The field *keypath* allows the application to identify each entry with an unique number. Creation of indizes closes the *openDatabase* function. An index is required to query the object store for a specific attribute, e.g. a calling telephone number. The remaining functions *onsuccess* (line 29) and *onerror* (line 34) are callback functions and handles success and errors of database opening. Depending on the state, the deferred is resolved (success, line 31) or rejected (failure, line 36). The return of the promise in line 38, which is called immediately after requesting to open the database *metadata* (because *open* is an asynchronous function and the call of *return* is the next function call in the *openDatabase* function).

To deal with the blocking behaviour of the database actions the other functions as well work with the Promise API.

The key part of the *selectData* function (see listing 4.9) opens a readonly transaction (line 1). To access the stored values, the instantiation of a cursor is required (line 5). This action triggers the *onsuccess* function (line 7) and passes an parameter containing the selected objects from the object store. While there are no restrictions defined, the IndexedDB returns all values of the datastore. As long as data is available, the result is pushed into the *returnValue* array and returned via promise when all values are read (line 10).

```
1  var trans = db.transaction(["metadata"], "readonly");
2  var store = trans.objectStore("metadata");
3  var returnValue = [];
4  var cursorRequest = store.openCursor(); // request cursor
5  cursorRequest.onsuccess = function(e) {
6    var result = e.target.result;
7    if(result === null || result === undefined) {
8      deferred.resolve(returnValue); // return values
9    } else {
10     returnValue.push(result.value);
11     result.continue();
12   }
13 };
```

Listing 4.9: Selecting data from the IndexedDB

Finally, the *resetDatabase* function (see listing 4.10) retrieves the object store *metadata* and clears it (line 3).

```
1  var trans = db.transaction(["metadata"], "readwrite");
2  var store = trans.objectStore("metadata");
3  var request = store.clear();
```

Listing 4.10: Reset object store

The functions *updateData* and *deleteData* are not used at the moment.

4.2.6 D3.js

The D3.js library is responsible for the visualisation of the provided data and is embedded with the usage of an AngularJS directive. This section describes the directive and implementation for the bar-chart only, because the other charts/graphs follow the same paradigms. The AngularJS directive in the HTML5 part is a placeholder for a more extensive behavior of a component. While HTML is case-insensitive, AngularJS converts dash-delimited descriptors to camel-case. The following *bar-chart* HTML5 component is referenced in AngularJS as *barChart* (see listing 4.11).

```
1  <bar-chart service="TEL" data="data"></bar-chart>
```

Listing 4.11: HTML code of an AngularJS directive

```
1  app.directive('barChart', function() {
2    var chart = d3.custom.barChart();
3    return {
```

```
4     restrict: 'E',
5     replace: true,
6     scope:{
7         data: "=",
8         service: "="
9     },
10    template: '<div class="chart"></div>',
11    link: function(scope, element, attrs) {
12      var chartEl = d3.select(element[0]);
13      scope.$watch('data', function (newVal, oldVal) {
14        if (newVal !== undefined) {
15          chartEl.datum({data: newVal, service: attrs.service}).call(
                chart);
16        }
17      });
18    } // end link function
19  }; // end return
20 });
```

Listing 4.12: Bar-chart directive

The *barChart* directive (see listing 4.12) creates an *div* HTML5 element with a class attribute containing *chart* (line 10). This class identifier is used to format the graph with the help of CSS. The *restrict* parameter in line 4 forces the directive to match in the DOM with the name of the element only. Other possibilities are A (match attribute name) and C (match class name). *Replace* (line 5) forces the directive to overwrite the DOM entry with the new code of the template (line 10). While AngularJS depends on Dependency Injection (DI), the scope of the directive can be filled with external data. In this case, each bar-chart gets the user provided data and the service to filter the data. Finally, the *link* attribute (line 11) contains the function, which enriches the DOM element with new behaviour. As soon as the *data* element changes, the watch function is triggered and the visualisation is called.

The description of the visualisation part of the directive (see listing 4.13) is shortened by some lines for better oversight. The D3.js API is addressed by the keyword *d3*. The bar-chart is stored in the custom area of the D3.js implementation. The first lines of the module defines the size and spacing of the chart (line 4-6). Beginning with line 8, the function responsible for the chart creation is declared. While the data provided to the bar-chart consists of rows containing from, to, date/time and duration, the data must be summarised by date to create bars over a time axis, the x-axis.

```
1  d3.custom = {};
2
3  d3.custom.barChart = function module() {
4    var margin = {top: 10, right: 10, bottom: 100, left: 100};
5    var width = 960, height = 500, gap = 0, ease = 'cubic-in-out',
         duration = 500;
6    var svg;
7
8    function chart(_selection) {
9      _selection.each(function (_data) {
10       var nested_data = d3.nest()
11           .key(function(d) { return d.date; }).sortKeys(d3.ascending)
12           .key(function(d) { return d.service; })
```

```
13          .rollup(function(d) {
14            return d3.sum(d, function(g) {
15              return g.duration;
16            })
17          })
18          .entries(_data);
19
20      // rebuild array of objects
21      nested_data.forEach(function(d) {
22          d.date = d.key;
23          d.values.forEach(function(e) {
24            d[e.key] = e.values;
25          });
26          delete(d.key);
27          delete(d.values);
28      });
29
30      var chartW = width - margin.left - margin.right,
31          chartH = height - margin.top - margin.bottom;
32
33      var parseDate = d3.time.format("%Y-%m-%d").parse;
34
35      var x = d3.scale.ordinal()
36          .rangeRoundBands([0, chartW], .05);
37
38      var y = d3.scale.linear()
39          .domain([0, d3.max(nested_data, function(d, i){ return d; })
                ])
40          .range([chartH, 0]);
41
42      var xAxis = d3.svg.axis()
43          .scale(x)
44          .orient("bottom")
45          .tickFormat(d3.time.format("%Y-%m-%d"));
46
47      var yAxis = d3.svg.axis()
48          .scale(y)
49          .orient("left")
50          .ticks(10);
51
52      if (!svg) {
53          svg = d3.select(this).append("svg").classed("chart", true);
54          var container = svg.append("g").classed("container-group",
                true);
55          container.append("g").classed("chart-group", true);
56          container.append("g").classed("x-axis-group axis", true);
57          container.append("g").classed("y-axis-group axis", true);
58      } // if !svg
59
60      x.domain(nested_data.map(function(d) { return parseDate(d.date);
            }));
61      y.domain([0, d3.max(nested_data.filter(function(value) {
62          return typeof value === "number";
63      }))]);
64
65      var bars = svg.select('.chart-group')
```

```
66            .selectAll('.bar')
67            .data(nested_data);
68
69        bars.enter().append('rect')
70            .classed('bar', true)
71            .attr("x", function(d) { return x(parseDate(d.date)); })
72            .attr("width", x.rangeBand())
73            .attr("y", function(d) {
74              if (d.hasOwnProperty("TEL")) {
75                  return y(d.TEL);
76              } else {
77                  return 0;
78              }
79            })
80            .attr("height", function(d) {
81              if (d.hasOwnProperty("TEL")) {
82                return height - y(d.TEL);
83              } else {
84                return 0;
85              }
86            });
87      });
88    };
89    return chart;
90  };
```

Listing 4.13: Bar-chart D3.js implementation

First, the *d3.nest()* function (line 10-18) is used to group the call history hierarchically by date and service and sort them ascending by date (line 11, 12). In line 14, the duration of each service to each date is summarised and returned. In a second step, the data is now standardised for further processing (see listing 4.14).

In line 35 and line 38 the scales for the x and y axis of the graph are defined. D3.js describes scales as functions that map from an input domain to an output range. The x-axis gets an ordinal scale, in this case a fixed number of dates, whereas the y-axis gets a linear scale ranging from the height of the chart to 0 (zero). The inverted order derives from the fact, that SVG has the origin of its coordinate system in the top left corner. Therefore, the highest bar has the lowest gap to the top of the graph. This is reflected in the reverse domain/range relationship (line 39 and 40). The *rangeRoundBands* function of the x-axis ensures, that the bars are aligned properly. The length of the x-axis is divided by the number of days in the user's data which are rounded to the nearest whole number. In line 53-57, the SVG element is created and some containers for grouping the bars and axes are appended. Line 60 and 61 provide the data to the axes. The bars are created by the lines 65-67. For each object in *nested_data*, one bar is appended to the chart. The layout of the bar takes place in the lines 69-86 where the x and y positions as well as the height and width are set.

The charts are based on the data provided by the user's file upload and has to be normalised for the visualisation component (see listing 4.14).

```
1 // data passed to directive
2 Array [
3 Object {datavolume: 0, date: "2013-12-20", duration: 1, from: "
    06802072XXX", id: 743, service: "SMS", source: "
```

```
    EVN_1401_518225XXX_1_06802072XXX.csv", time: "10:12:34", timestamp
    : 1387530754000, to: "06768520XXX"},
4 Object {datavolume: 0, date: "2013-12-20", duration: 1, from: "
    06802072XXX", id: 744, service: "SMS", source: "
    EVN_1401_518225XXX_1_06802072XXX.csv", time: "12:54:46", timestamp
    : 1387540486000, to: "06768520XXX"},
5 Object {datavolume: 11904, date: "2013-12-26", duration: 0, from: "
    06802072XXX", id: 745, service: "DATA", source: "
    EVN_1401_518225XXX_1_06802072XXX.csv", time: "00:00:00", timestamp
    : 1388012400000, to: "BOB.AT"}, ...]
6
7 // data after d3.nest()
8 Array [
9 Object {key: "2013-12-20", value: Object {key: "service", value: Array
     [ Object{key: "TEL", value: "300"}, Object{key: "SMS", value: "5"
    }]}},
10 Object {key: "2013-12-26", value: Object {key: "service", value: Array
     [ Object{key: "DATA", value: "22464"}]}},
11 Object {key: "2013-12-27", value: Object {key: "service", value: Array
     [ Object{key: "DATA", value: "5666"}, Object{key: "TEL", value: "
    1127"}}}, ...]
12
13
14 // data after forEach normalisation
15 Array [
16 Object {DATA: 0, SMS: 5, TEL: 300, WHATSAPP: 0, date: "2013-12-20"},
17 Object {DATA: 22464, SMS: 0, TEL: 0, WHATSAPP: 0, date: "2013-12-26"},
18 Object {DATA: 5666, SMS: 0, TEL: 1127, WHATSAPP: 0, date: "2013-12-27"
    }, ...]
```

Listing 4.14: Data processing for bar-chart

The bar-chart and the stream-graph implements an mouse over effect too. When the mouse pointer moves over a bar or a stream, the duration and the target of the communication is displayed.

4.2.7 Whatsapp Database Import

Whatsapp (see section 2.6.1) allows its users to create backups of their message history (automatically each day at 04:00 am). These backups are stored in an SQLite[26] database format and could be accessed with free SQLite tools[27].

Nevertheless, for higher security, these backups are encrypted to prevent unauthorised access. Therefore, to process the message history, Whatsapp's encryption has be removed. We have seen two different encryption methods for Whatsapp backups so far:

- CRYPT5

[26] SQLite is the most widely deployed SQL database engine in the world `http://www.sqlite.org/`, last seen on 10th July 2014

[27] e.g. SQLite browser `http://sourceforge.net/projects/sqlitebrowser/`, last seen on July, 10 2014

- CRYPT7

While CRYPT5 could be decrypted with the leaked keys (identical initialization vector for all users) and the encrypted database file, the only tool needed is OpenSSL[28] , the current version of Whatsapp uses their CRYPT7 algorithm in combination with a server derived key. This key can change on a regular base and is not accessible for non-root users. Therefore, a full backup of the Android phone is required.

Extracting CRYPT5 database

Locate the Whatsapp CRYPT5 database file (see listing 4.15) on an Android based phones and copy it to a computer.

```
1 /sdcard/WhatsApp/Databases/msgstore.db.crypt5
```

Listing 4.15: Path to the user accessible CRYPT5 message history backup

The CRYPT5 algorithm uses the associated Google account of your phone to encrypt the file. So, open a shell and create a MD5 hash of the Google mail address (see listing 4.16):

```
1 echo -n abcd@gmail.com | md5sum
2 46040c38d1cbe8ffcd3df6c8ba787951 *-
```

Listing 4.16: Creating MD5 hash of your E-Mail address

This 32-digit (128 bit) account hash must be extended to 48-digits (192 bit) by adding the first 16 digits of the original hash (see listing 4.17):

```
1 46040c38d1cbe8ffcd3df6c8ba787951 --> // 32-digit account hash
2 46040c38d1cbe8ffcd3df6c8ba78795146040c38d1cbe8ff // 48-digit account
    hash
```

Listing 4.17: Extending the 32-digit hash to 48-digits

To retrieve the key for decryption of the database file, XOR the 48-digit key with the following static key (see listing 4.18):

```
1 8d4b155cc9ff81e5cbf6fa7819366a3ec621a656416cd793 // static key
2 XOR
3 46040c38d1cbe8ffcd3df6c8ba78795146040c38d1cbe8ff // 48-digit account
    hash
4 --> KEY
```

Listing 4.18: XOR operation with two keys

Finally, use the following initialisation vector (IV) 1e39f369e90db33aa73b442bbbb6b0b9 to decrypt the database (replace [key] and [iv] with the corresponding values) (see listing 4.19):

```
1 openssl enc -aes-192-cbc -d -nosalt -in msgstore.db.crypt5 -out
    msgstore.db -K [key] -iv [iv]
```

Listing 4.19: Decrypt CRYPT5 message history backup with OpenSSL

[28] Free toolkit implementing Secure Sockets Layer (SSL) and Transport Sockets Layer (TLS) `http://www.openssl.org/`, last seen on 10th July 2014

Now, the decrypted database file should be ready to open with any SQLite tool. A script[29] to decrypt the CRYPT5 database will be provided with the source code to the thesis.

Extracting CRYPT7 database

While the key for the CRYPT7 encryption is different from other user's key, it could not be leaked. Therefore, the key has to be retrieved from the phone's root partition. There are two possibilities to get the key:

- Root the phone and get access to the root partition or
- create a backup of the (interesting parts) of the root partition.

Rooting an Android phone differs between make and model, so we have chosen to create a backup of the phone. For the retrieval of the Whatsapp database from the phone's root partition the following tools are required:

- Phone with Android OS 4+ (codename: Ice-Cream-Sandwich)
- USB debugging enabled
- Java 7 JRE (unencrypted phone)
- Android SDK bundle[30] (32 or 64 bit depending on your CPU)
- OpenSSL (unencrypted phone)
- Java 7 JDK (encrypted phone)
- Bouncy Castle Provider (encrypted phone)
- Java Cryptography Extension (JCE) Unlimited Strength Jurisdiction Policy Files 7 (encrypted phone)

First, check if the phone has the required Android version installed (*Settings* > *About phone* > *Android version* (see figure 4.5). Then enable USB debugging (see figure 4.6): *Settings* > *Developer options* > *USB-Debugging*: enable. If the phone does not display *Developer options* on the lower end of the settings menu (above *About phone*), the menu has to be enabled first. Go to *Settings* > *About phone*, scroll down to *Build number* and tap the build number 7 times (see figure 4.5). You should see a message and finally the *Developer options* in the *Settings* menu.

Download and extract the Android SDK bundle. Change to the directory and start *SDK Manager.exe*. This application suggests some packages for application development, which are not required for our purpose. Click *Deselect all*, scroll down and select *Google USB Driver* (see figure 4.7). Click *Install 1 package...*. A new window opens (see figure 4.8) where you have to click on the Google USB Driver entry and accept the licence to download and install the driver.

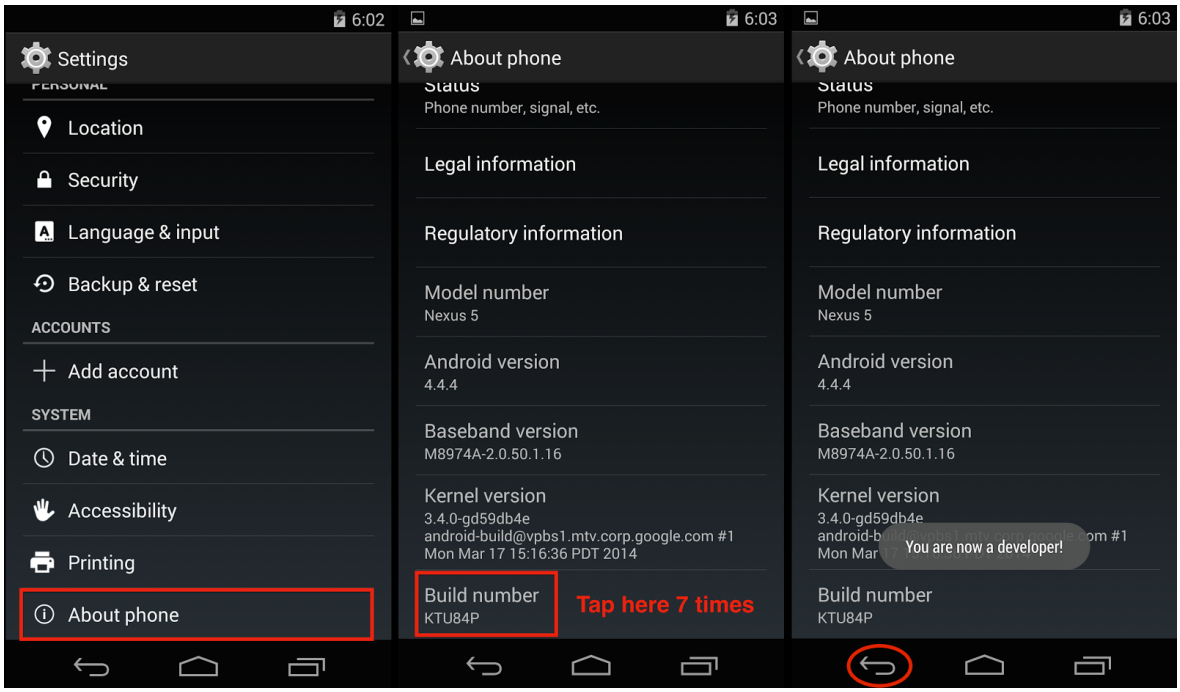


Figure 4.5: Android Settings menu

If the driver installed successfully, connect the Android phone to the USB port of the PC. The PC should recognize the phone and display a connected device icon in the system tray.

Now open a shell in the *<AndroidSDKbundle>/sdk/platform-tools/* directory and execute the following command (see listing 4.20):

```
1 adb backup -noshared -nosystem com.whatsapp
```

Listing 4.20: Using Android Debug Bridge (ADB) to create a backup of the phone

The PC connects to the phone and asks for permission to backup its data. On the phone, the PC has to be authorized (see screenshot 4.9).

After permitting the backup, the PC copies the data and stores these into *backup.ab* in the current directory. The process duration depends on the size of the message database. With the direct selection of the Whatsapp package, the process should only run some seconds and create some MB of storage (compressed message history).

Finally, the backup process finished and created a *backup.ab* file with 2.5 MB on the disk. Depending on the encryption status of the phone (encrypted/unencrypted), the file header can have two different values. The unencrypted header looks like (see listing 4.21):

```
1 ANDROID BACKUP
2 1
3 1
4 none
```

[29] see `http://www.digitalinternals.com/279/20140330/decrypt-whatsapp-crypt5-database-messages/`, last seen on 10th July 2014

[30] `http://developer.android.com/sdk/index.html#download`, last seen on 10th July 2014

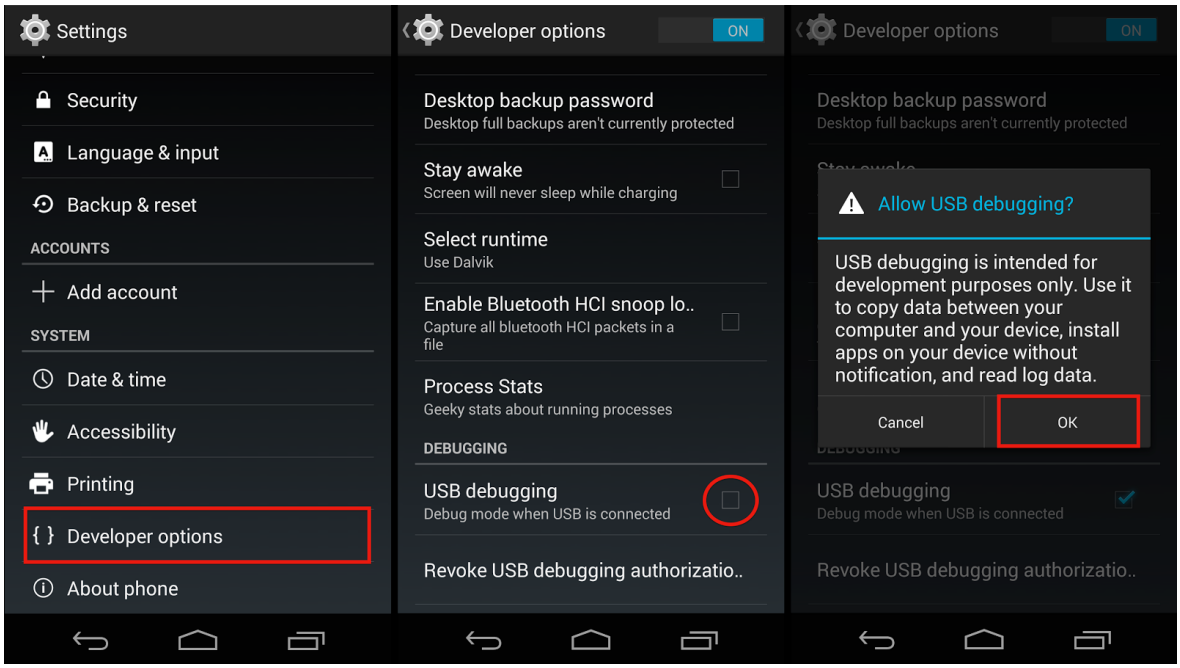


Figure 4.6: Android Settings menu

```
5 <data>
```

Listing 4.21: Header of plain android backup

Whereas the encrypted header contains additional information (see listing 4.22:

```
1  ANDROID BACKUP
2  1
3  1
4  AES-256
5  3A2155B...hexadecimal user password salt
6  430EB9C...hexadecimal master key checksum salt
7  10000     number of PBKDF2 rounds
8  43DAA1E...hexadecimal user key initialization vector
9  90D9E43...hexadecimal master key
10 <data>
```

Listing 4.22: Header of encrypted android backup

The unencrypted backup is compressed with the deflate algorithm which can be reversed by executing the following commands (see listing 4.23):

```
1 dd if=backup.ab bs=24 skip=1|openssl zlib -d > backup.tar
2 tar xvf backup.tar
```

Listing 4.23: Extract unencrypted but compressed CRYPT7 message history

As a result, the files from the backup file are restored in the directory and can be accessed by the user.

The encrypted backup needs additional steps to gain access to the message history. To encrypt the file system, Android generates an AES256 key from the user supplied

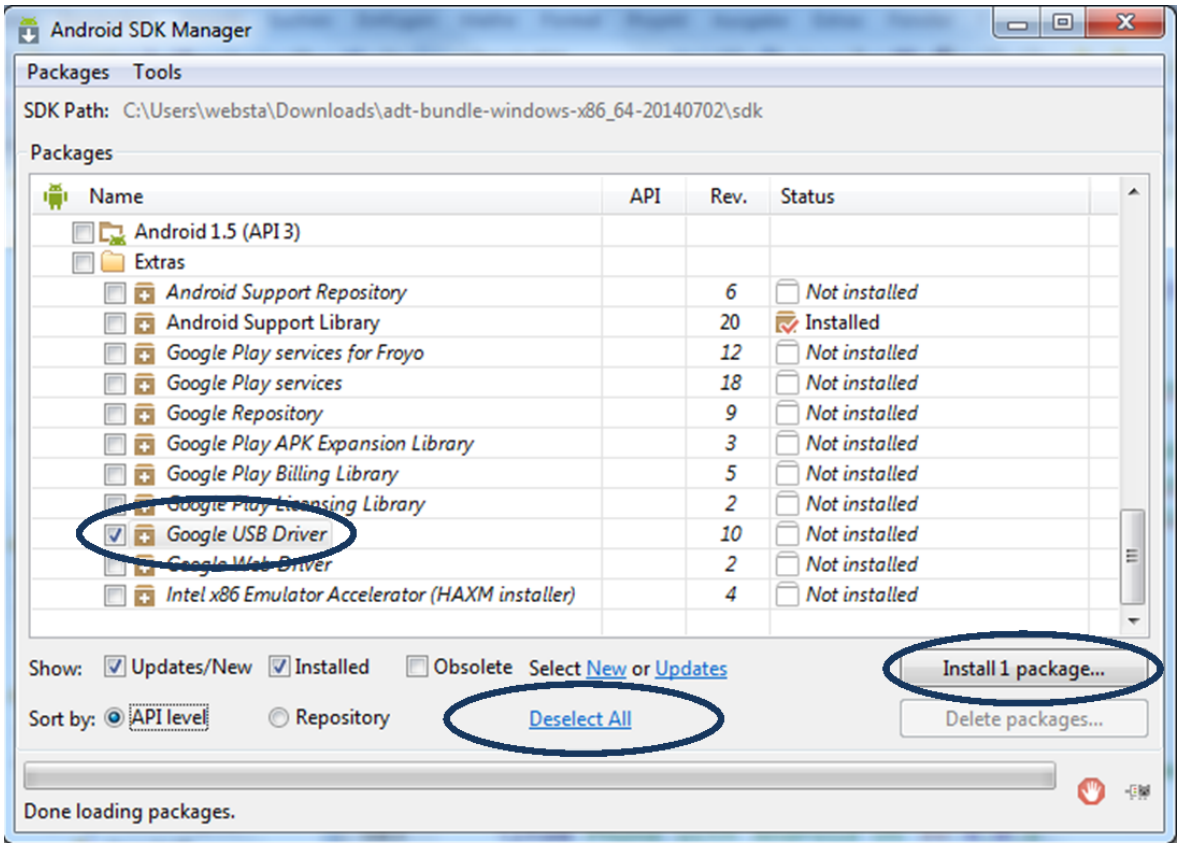


Figure 4.7: Android SDK Manager screen (step 1)

password using 10.000 rounds of PBKDF2[31] with a random, 512 bit salt. A randomly generated master key will be AES256 encrypted with this key. Finally, the master key encrypts the file system in CBC mode. For the decryption of the backup we use a Java application called **Android-Backup-Extractor** (ABE) which is available on GitHub[32]. To run this application, an additional library called **Bouncy Castle Provider** (bcprov-jdk15on-151.jar) is required (available for free download[33]). Put the file into the lib directory of the ABE. The last requirement, the JCE unlimited strength jurisdiction policy files for Java 7 are available for free download[34] too (accept license agreement). Extract the package and place the file **US_export_policy.jar** into <jdk7>/jre/lib directory. Finally start the ABE by executing the following command containing the encryption password (XXXX) in the ABE root directory (see listing 4.24):

```
1 java -cp :bin:lib/bcprov-jdk15on-151.jar org.nick.abe.Main unpack
    encrypted-backup.ab decrypted-backup.tar XXXX
```

Listing 4.24: Using Android Backup Extractor (ADB) to decrypt the phone backup

[31] Password-Based Key Derivation Function 2, part of Public-Key Cryptography Standards (PKCS), RFC2859

[32] `https://github.com/nelenkov/android-backup-extractor`, last seen on 10th July 2014

[33] `http://www.bouncycastle.org/latest_releases.html`, last seen on 10th July 2014

[34] `http://www.oracle.com/technetwork/java/javase/downloads/jce-7-download-432124.html`, last seen on 10th July 2014

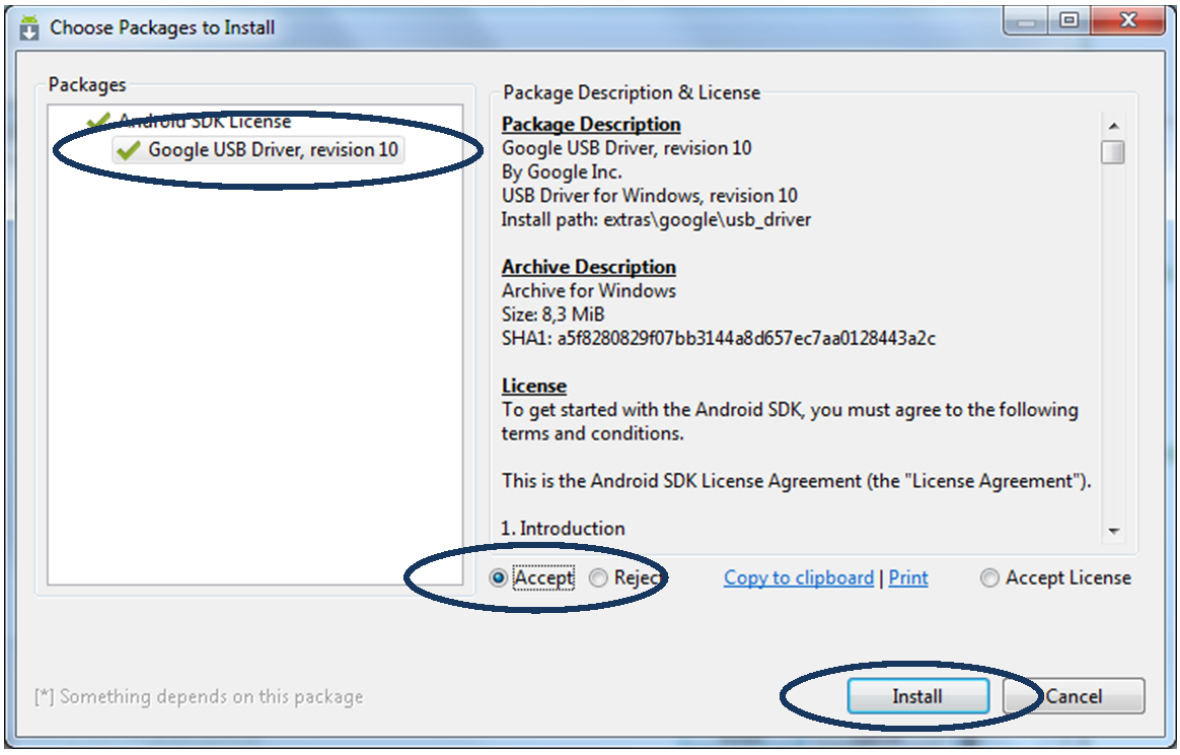


Figure 4.8: Android SDK Manager screen (step 2)

As a result, the backup of the Android phone should exist decrypted in this directory and can be extracted by calling (see listing 4.25):

```
1 tar xvf decrypted-backup.tar
```

Listing 4.25: Extract files from tape archiver (TAR) file

The extraction process creates the file structure of the Whatsapp application on the PC as it exists on the Android phone (see listing 4.26):

```
1  apps
2  apps/com.whatsapp
3  apps/com.whatsapp/db
4  apps/com.whatsapp/db/msgstore.db
5  apps/com.whatsapp/db/msgstore.db-journal
6  apps/com.whatsapp/db/wa.db
7  apps/com.whatsapp/db/wa.db-shm
8  apps/com.whatsapp/db/wa.db-wal
9  apps/com.whatsapp/f
10 apps/com.whatsapp/f/.trash
11 apps/com.whatsapp/f/account_type
12 apps/com.whatsapp/f/Avatars
13 apps/com.whatsapp/f/Avatars/4365012345678@s.whatsapp.net.j
14 apps/com.whatsapp/f/Avatars/4365023456789@s.whatsapp.net.j
15 ...
16 apps/com.whatsapp/f/emoji
17 apps/com.whatsapp/f/expiration_date
18 apps/com.whatsapp/f/fullsync.dat
19 apps/com.whatsapp/f/full_sync_wait
20 apps/com.whatsapp/f/invalid_numbers
```

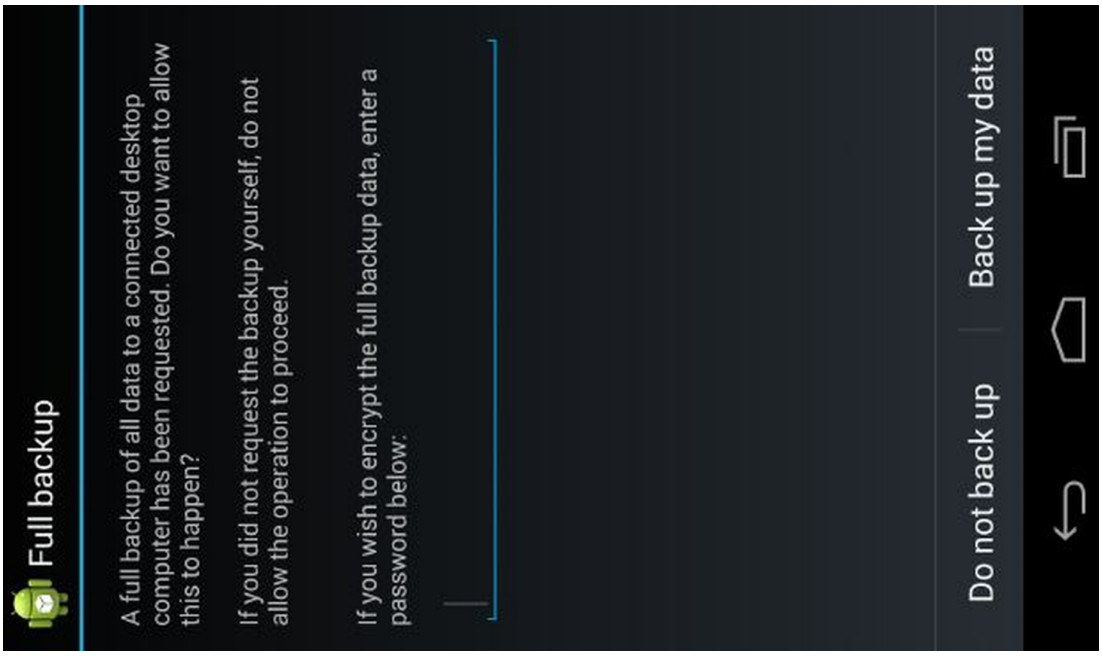


Figure 4.9: Android full backup permission

```
apps/com.whatsapp/f/key
apps/com.whatsapp/f/Logs
apps/com.whatsapp/f/Logs/whatsapp-2014-0005-01.1.log.gz
apps/com.whatsapp/f/Logs/whatsapp-2014-07-24.1.log.gz
apps/com.whatsapp/f/Logs/whatsapp-2014-07-25.1.log.gz
apps/com.whatsapp/f/Logs/whatsapp-2014-07-26.1.log.gz
apps/com.whatsapp/f/Logs/whatsapp.log
apps/com.whatsapp/f/me
apps/com.whatsapp/f/MessageService.pid
apps/com.whatsapp/f/pw
apps/com.whatsapp/f/rc
apps/com.whatsapp/f/statistics
apps/com.whatsapp/f/sync_backoff
apps/com.whatsapp/f/wastats.dims
apps/com.whatsapp/f/wastats.log
apps/com.whatsapp/f/wastats.timestamp
apps/com.whatsapp/r
apps/com.whatsapp/r/app_Keys
apps/com.whatsapp/r/app_webview
apps/com.whatsapp/r/app_webview/Web Data
apps/com.whatsapp/r/app_webview/Web Data-journal
apps/com.whatsapp/sp
apps/com.whatsapp/sp/com.whatsapp_preferences.xml
apps/com.whatsapp/sp/RegisterPhone.xml
apps/com.whatsapp/sp/VerifySms.xml
apps/com.whatsapp/sp/_has_set_default_values.xml
apps/com.whatsapp/_manifest
```

Listing 4.26: Listing of Whatsapp directory structure

The file structure contains the encryption key for the encrypted Whatsapp message history database (*apps/com.whatsapp/f/key*) which can be used in several applications

to decrypt the user accessible backup directly on the phone. Nevertheless, much more interesting is the unencrypted message database (*apps/com.whatsapp/db/msgstore.db*) which saves us from decrypting the backup database. So the further investigation of Whatsapp is based on this SQLite database.

4.2.8 File Format description

While different TSP/ISP provides itemised bills in CSV file format or in PDF[35], the implementation has to provide several import implementations. Depending on the selected format in the UI, the parsing component switches to the appropriate implementation to to extract the required information. For CSV files, the implementation contains a bunch of regular expressions to read the required lines from a file. So, inappropriate lines have to be skipped (e.g. headers and summaries). While some mature CSV parsing implementations are available (e.g. Papa Parse[36], CSV-2-array[37] or d3.csv[38], last seen on 10th July 2014), none of these is flexible enough to skip non-matching lines. So this implementation splits lines at \r\n (Windows CR & LF) or \n (Unix LF) and columns are split at semicolon (;). Finally, each extracted row is checked for some characteristics (number of columns and column format), qualifying it as a correct input row. Nevertheless, the user has to take care to select the correct provider in the UI.

The parsing algorithm is encapsulated in the *ParseFactory* which is part of the services.

BOB - CSV File Format

The itemised bill from BOB starts with some lines containing information about the user, billing details and the description of the following data lines (see listing 4.27).

```
1  Einzelverbindungsnachweis
2  Kundennummer: 518XXXXXX/1
3  Rechnungsnummer: 500051XXXXXX
4  Rechnungsdatum: 29.11.2013
5  Periode: 11/2013
6  Abrechnungszeitraum: 26.10.2013-25.11.2013
7  Faelligkeit: 03.12.2013
8  Rufnummer;Datum;Beginn;Service;Dauer;Taktung;Netzbetreiber;Zone/Typ;
       Zielrufnummer;Upload (kB);Download (kB);Volumen gerundet (kB);
       Netto EUR
9  06802072XXX;26.10.2013;00:00:00;UMTS;24:00:00;00:00:00;Oesterreich  -
       bob;Datenvolumen GPRS/UMTS;BOB.AT;472,59;4766,69;5248,00;0,0000
10 06802072XXX;26.10.2013;14:15:36;TEL;00:00:05;00:01:00;Oesterreich  -
       bob;andere Mobilnetze;00436505800XXX;;;;0,0000
```

[35]Portable Document Format

[36]Papa is a powerful jQuery plugin that parses CSV (delimited text) input and can handle large files by streaming them `http://papaparse.com/`, last seen on 10th July 2014

[37]A compact (645 bytes) but compliant function to convert a CSV string into a 2D array, conforming to the RFC4180 standard `https://code.google.com/p/csv-to-array/`, last seen on 10th July 2014

[38]see `https://github.com/mbostock/d3/wiki/CSV`

```
11 06802072XXX;26.10.2013;15:34:14;SMS;00:00:00;00:00:00;Oesterreich  -
      bob;SMS gesendet;06766335XXX;;;;0,0000
```

Listing 4.27: Excerpt of BOB itemised bill

The advantage of these files is, that all services (telephony, SMS and mobile data) uses the same format. Depending on the service, some columns are empty whereas others are filled. According to listing 4.27, a qualifying row has 13 columns, which is checked in a loop iterating over the rows. Finally, BOB uses the letter *X* to make anonymous the callee's number.

TELERING - CSV File Format

Telering uses a very similar format, except the number of columns is reduced to 9 (see listing 4.28).

```
1 Rufnummer;Datum;Uhrzeit;Dauer;Zielrufnummer;Betrag Netto;Zone;Download
      (KB);Upload (KB)
2 0650 5800XXX;01-04-14;08:14:08;16:04:08;web;0,0000;Internet;8186;1279
3 0650 5800XXX;01-04-14;11:19:26;---;+43676898599XXX;0,0000;SMS extern;;
4 0650 5800XXX;01-04-14;18:29:12;00:02:39;+436642819XXX;0,0000;A1;;
```

Listing 4.28: Excerpt of TELERING itemised bill

YES - CSV File Format

Yes is the first provider, which separates the mobile data usage (lower end of the CSV file) from telephony and SMS by an empty line (see listing 4.29).

```
1 Datum;Uhrzeit;Von;Zielrufnummer;Service-Beschreibung;Dauer;Netto/EUR
2 26.04.2014;10:42:51;4368181875XXX;436509550***;SMS;00:00:00;0,0000
3 26.04.2014;18:35:24;4368181875XXX;436509550***;Telefonie Oesterreich
     ;00:00:37;0,0000
4 27.04.2014;12:07:42;4368181875XXX;43820810***;Preisregulierter Dienst
     ;00:01:48;0,2333
5
6 26.04.2014;08:25:19;4368181875XXX;Oesterreich;Datentransfer: 72,0 KB
     ;00:00:06;0,0000
7 26.04.2014;09:07:19;4368181875XXX;Oesterreich;Datentransfer: 64,0 KB
     ;00:00:20;0,0000
```

Listing 4.29: Excerpt of YES itemised bill

Another difference is the character * which is used by Yes to make the callee's number anonymous (see listing 4.29). Nevertheless, the parsing factory uses its own code to make numbers anonymous, which includes the replacement of * by *X*. This standardisation allows the implementation to match the numbers of different itemised bills.

Drei - PDF

After parsing CSV files, Drei[39] does provide itemised bills in portable document format. Additionally to the format, Drei splits its itemised bill into several sections (see figure 4.10).

Kundennummer	Rechnungsnummer	Rechnungsdatum
9275[redacted]	6761[redacted]	25. März 2014

Detailübersicht für 066054[redacted]

Sprachtelefonie national (sekundengenaue Auflistung der Nutzung)

Datum	Zeit	Beschreibung	Telefonnummer	Dauer h:m:s	Netto
Di 25. Feb.	17:39:35	3	00436505702***	9:38	0,000
Do 27. Feb.	14:43:49	A1	00436642005***	0:07	0,000

SMS National

Datum	Zeit	Beschreibung	Telefonnummer	Anzahl	Netto
Sa 22. Feb.	14:38:24	SMS	00436642005***	1	0,000
Sa 22. Feb.	15:00:52	SMS	00436642005***	1	0,000

SMS zu 3

Datum	Zeit	Beschreibung	Telefonnummer	Anzahl	Netto
Fr 28. Feb.	19:07:07	SMS zu 3	004369911895***	1	0,000
Sa 1. März	07:32:30	SMS zu 3	00436505702***	1	0,000

Datendienste

Datum	Beginn	Dauer	Abrechnungsnummer	Beschreibung	Download	Upload	Netto
Sa 22. Feb.	10:50:12	14:26:57	205564[redacted]	Internet Dienste	1,94 MB	0,90 MB	0,000
So 23. Feb.	10:31:48	13:53:03	145262[redacted]	Internet Dienste	5,03 MB	3,94 MB	0,000

Figure 4.10: Excerpt of Drei itemised bill

To extract the content of the PDF on the client-side, the Mozilla library PDF.js[40], released under the Apache licence[41] is used. The library provides a function *getTextContent()* and delivers an array of strings, containing each text block of one page, line after line, blocks from left to right. So, the implementation has to iterate over the supplied files, over each page of each file and finally, over each block of text of each page to find some patterns identified as a relevant part of the itemised bill. For this purpose, some sequent text blocks where identified and used as pattern. This algorithm is implemented in the *ParseFactory* and extracts reliable all connection information. Drei especially, uses abbreviations of month names, if the name is longer than 4 characters. This does not correspond to the 3 character abbreviation standard the library D3.js understands. D3.js is used to parse and format the date and therefore a translation between German 3-letter with point (e.g. Feb.) and 4-letter (e.g. März) is required. Furthermore, the date does not include any year information. So, parsing of the itemised bill with date entries for December/January required additional coding.

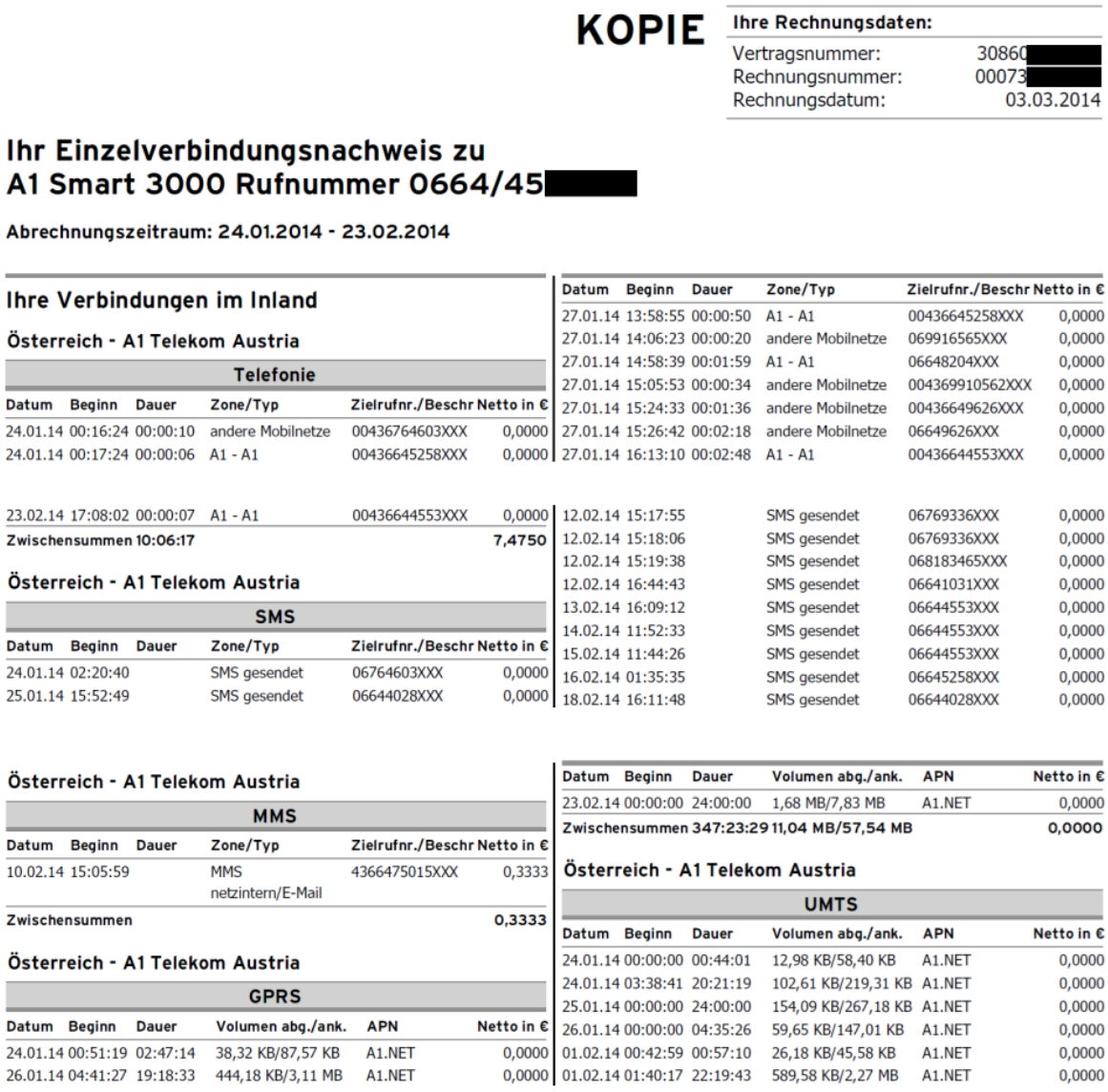
[39] Drei is the product of a merger on the Austrian telecommunications market between Drei and Orange `http://diepresse.com/home/techscience/mobil/1442727/Mobilfunk_Marke-Orange-verschwindet-endgultig`, last seen on 3rd August 2014

[40] PDF.js `http://mozilla.github.io/pdf.js/`, last seen on 3rd August 2014

[41] Apache licence `http://opensource.org/licenses/Apache-2.0`, last seen on 3rd August 2014

A1 - PDF

A1 too was not able to deliver a CSV file, even though A1 provided it in the past. Nevertheless, we could manage to get a PDF itemised bill which starts with an overview presenting costs for the contractual services. Not before page 3, the connection information is listed (see figure 4.11).



KOPIE

Ihre Rechnungsdaten:

Vertragsnummer: 30860[redacted]
Rechnungsnummer: 00073[redacted]
Rechnungsdatum: 03.03.2014

Ihr Einzelverbindungsnachweis zu
A1 Smart 3000 Rufnummer 0664/45[redacted]

Abrechnungszeitraum: 24.01.2014 - 23.02.2014

Ihre Verbindungen im Inland

Österreich - A1 Telekom Austria

Telefonie

Datum	Beginn	Dauer	Zone/Typ	Zielrufnr./Beschr	Netto in €
24.01.14	00:16:24	00:00:10	andere Mobilnetze	00436764603XXX	0,0000
24.01.14	00:17:24	00:00:06	A1 - A1	00436645258XXX	0,0000
23.02.14	17:08:02	00:00:07	A1 - A1	00436644553XXX	0,0000
Zwischensummen		**10:06:17**			**7,4750**

Österreich - A1 Telekom Austria

SMS

Datum	Beginn	Dauer	Zone/Typ	Zielrufnr./Beschr	Netto in €
24.01.14	02:20:40		SMS gesendet	06764603XXX	0,0000
25.01.14	15:52:49		SMS gesendet	06644028XXX	0,0000

Datum	Beginn	Dauer	Zone/Typ	Zielrufnr./Beschr	Netto in €
27.01.14	13:58:55	00:00:50	A1 - A1	00436645258XXX	0,0000
27.01.14	14:06:23	00:00:20	andere Mobilnetze	069916565XXX	0,0000
27.01.14	14:58:39	00:01:59	A1 - A1	06648204XXX	0,0000
27.01.14	15:05:53	00:00:34	andere Mobilnetze	004369910562XXX	0,0000
27.01.14	15:24:33	00:01:36	andere Mobilnetze	00436649626XXX	0,0000
27.01.14	15:26:42	00:02:18	andere Mobilnetze	06649626XXX	0,0000
27.01.14	16:13:10	00:02:48	A1 - A1	00436644553XXX	0,0000
12.02.14	15:17:55		SMS gesendet	06769336XXX	0,0000
12.02.14	15:18:06		SMS gesendet	06769336XXX	0,0000
12.02.14	15:19:38		SMS gesendet	068183465XXX	0,0000
12.02.14	16:44:43		SMS gesendet	06641031XXX	0,0000
13.02.14	16:09:12		SMS gesendet	06644553XXX	0,0000
14.02.14	11:52:33		SMS gesendet	06644553XXX	0,0000
15.02.14	11:44:26		SMS gesendet	06644553XXX	0,0000
16.02.14	01:35:35		SMS gesendet	06645258XXX	0,0000
18.02.14	16:11:48		SMS gesendet	06644028XXX	0,0000

Österreich - A1 Telekom Austria

MMS

Datum	Beginn	Dauer	Zone/Typ	Zielrufnr./Beschr	Netto in €
10.02.14	15:05:59		MMS netzintern/E-Mail	4366475015XXX	0,3333
Zwischensummen					**0,3333**

Österreich - A1 Telekom Austria

GPRS

Datum	Beginn	Dauer	Volumen abg./ank.	APN	Netto in €
24.01.14	00:51:19	02:47:14	38,32 KB/87,57 KB	A1.NET	0,0000
26.01.14	04:41:27	19:18:33	444,18 KB/3,11 MB	A1.NET	0,0000
23.02.14	00:00:00	24:00:00	1,68 MB/7,83 MB	A1.NET	0,0000
Zwischensummen		**347:23:29**	**11,04 MB/57,54 MB**		**0,0000**

Österreich - A1 Telekom Austria

UMTS

Datum	Beginn	Dauer	Volumen abg./ank.	APN	Netto in €
24.01.14	00:00:00	00:44:01	12,98 KB/58,40 KB	A1.NET	0,0000
24.01.14	03:38:41	20:21:19	102,61 KB/219,31 KB	A1.NET	0,0000
25.01.14	00:00:00	24:00:00	154,09 KB/267,18 KB	A1.NET	0,0000
26.01.14	00:00:00	04:35:26	59,65 KB/147,01 KB	A1.NET	0,0000
01.02.14	00:42:59	00:57:10	26,18 KB/45,58 KB	A1.NET	0,0000
01.02.14	01:40:17	22:19:43	589,58 KB/2,27 MB	A1.NET	0,0000

Figure 4.11: Excerpt of A1 itemised bill

The itemised bill uses a two-column layout and splits the data into several sections: Telephony, SMS, MMS, GPRS and UMTS. Additionally, each section starts with a header and ends with a summary. PDF.js delivers all lines of the left column followed by the lines of the right column. Here, patterns were identified to trigger the beginning of the sections. Inside the sections, the layout does not change an the parsing process collects the required columns.

WHATSAPP - SQLITE Format

Whatsapp uses SQLite to store its messages on the filesystem. The retrieval of the database is described in section 4.2.7. To load the SQLITE file on client-side we use the library SQL.js[42] which is the original SQLite implementation in C compiled with Emscripten[43] to Javascript and released under MIT licence[44]. This library in combination with the FileReader API (see section 4.1.3 allows the implementation to load the file and execute SQL[45] queries against the message database (see listing 4.30). Figure 4.12 provides an overview of the Whatsapp message history database.

```
1 select key_remote_jid as 'to', length(data) as volume, timestamp from
    messages where key_from_me = 1 and media_mime_type is null;
```

Listing 4.30: SQL query for Whatsapp message retrieval

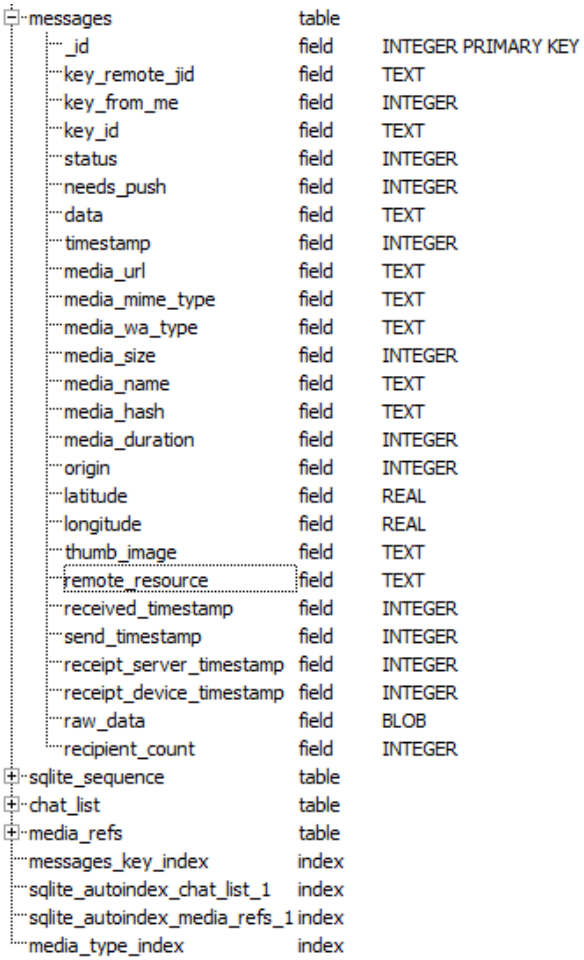


Figure 4.12: Structure of the Whatsapp message history database

[42] SQL.js `https://github.com/kripken/sql.js/`, last seen on 3rd August 2014

[43] An LLVM-to-JavaScript Compiler `https://github.com/kripken/emscripten`, last seen on 3rd August 2014

[44] MIT licence `http://opensource.org/licenses/MIT`, last seen on 3rd August 2014

[45] Structured Query Language

Chapter 5

Evaluation

5.1 Dataset description

This section describes the available communication dataset, which is used in section 5.2 for discussion and evaluation of the implementation.

We were able to acquire itemised bills from the following providers:

- A1 as PDF (1 user, 1 bill)
- Bob as CSV (1 user, 8 bills)
- Drei as PDF (1 user, 2 bills)
- T-Mobile as CSV (1 user, 2 bills)
- Telering as CSV (2 users, 2 bills)
- Whatsapp message history as SQLite Database (1 user, 1 database)
- Yes as CSV (1 user, 1 bill)

Some of the users have overlapping social networks, which is reflected in the itemised bills (e.g. mutual calls). The application allows users to import files from different providers step-by-step to enrich their dataset and benefit from the rising number of call record entries.

5.2 Discussion

This section describes the workflow required for visualisation of communication patterns.

With the help of D3.js (see section 4.1.5) as visualisation library, it is possible to create modern and responsive graphs in a browser environment. As much as important are the FileReader API and the IndexedDB which allowed the application to load files from the user and use the content on the client-side for visualisation. These components are crucial to guarantee privacy to users importing files into the application.

5.2.1 Scenario 1: Single-user and Single-source

In the first scenario, we analyse the content of one type of source from one user only (e.g. all itemised bills or the Whatsapp message history of one user). When the application was finally finished, itemised bills were loaded (on the left side of the application) and the resulting visualisation (right side) presented insight into the development team's communication behaviour. Some files showed that nearly all communication is done with the help of SMS, whereas others used the telephone in a way it was invented for: telephony. Furthermore, it is possible to import the Whatsapp message history to create a more complete view of the social network over a long period of time.

Not only the communication behaviour becomes visible (see figures 5.1, 5.2, 5.3, 5.4, 5.5, 5.6), but also the amount of time someone uses its mobile phone for telephony, SMS and data services. Reoccurring events visually strike down in the graphs and allows conclusions of one's life. One itemised bill revealed periodic communication activities on Fridays, caused by a bunch of sent SMS to a large number of friends (see figure 5.2).

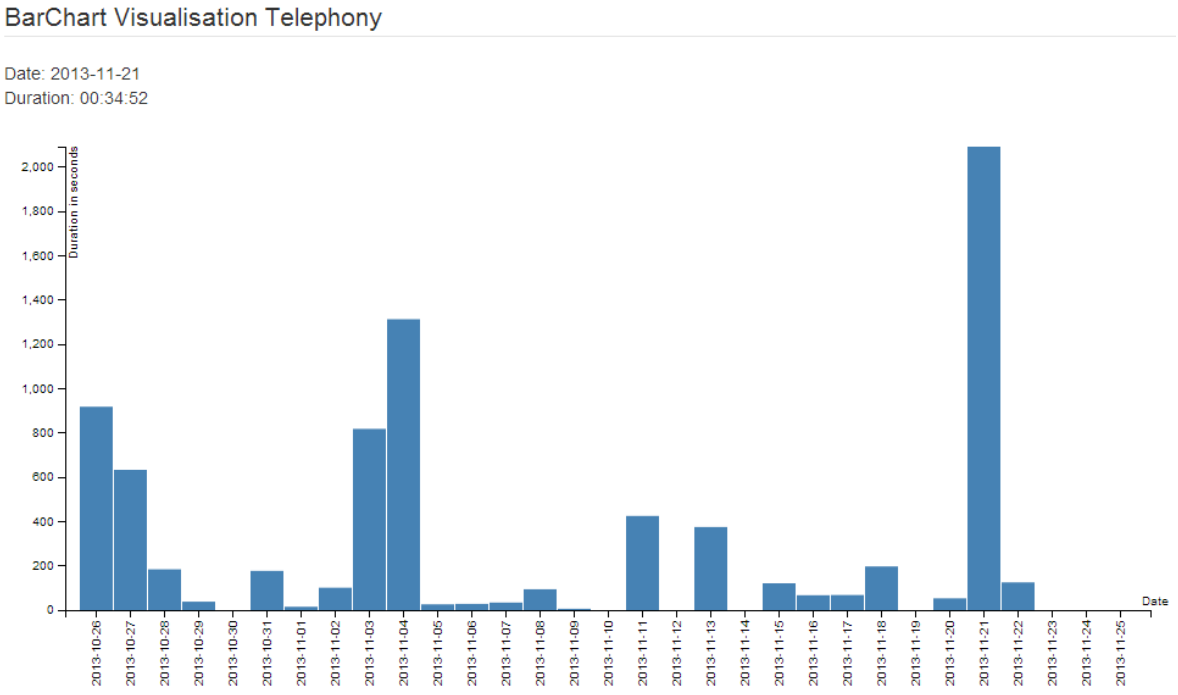


Figure 5.1: Screenshot of the bar-chart telephony visualisation

The bar-charts (see figures 5.1, 5.2, 5.3) and the stream-graph (see figures 5.4, 5.5, 5.6) supports a mouse over effect, that displays additional information in the top left corner of the graph. Bar-charts provide a good overview of the supplied data. The user can investigate the communication effort per day and can link these with past events. For better overview, the application displays one bar-chart for each type of communication: telephony, SMS and data. Another reason is the scale of the y-Axis. We did not find a reasonable scale to mix telephony seconds, SMS count and kilobytes together.

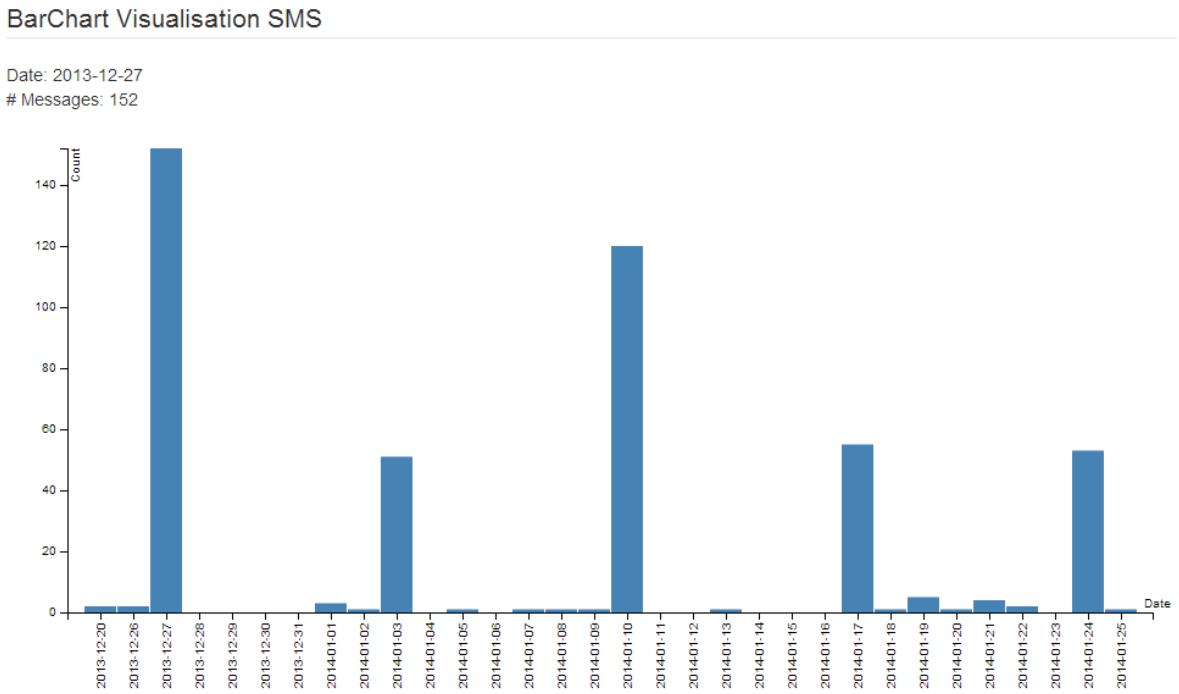


Figure 5.2: Screenshot of the bar-chart SMS visualisation

The stream-graphs are split into telephony, SMS and data too. Their advantage is the display of the whole communication effort (height of the wave) as well as the display of effort spent with one communication partner (relative thickness of one stream in the wave). On mouse-over, the underlying stream becomes highlighted, which alleviates investigation of the communication progress to one specific person.

Finally, the force-graph visualisation (see figure 5.7) representing the social network in star form, revealed close friends and distant relationships. To get the nodes in a reasonable ratio, the data services where removed from the force-graph for three reasons:

1. Data services always ends at the service provider.
2. The services are counted in kilobytes and pollute the scale.
3. No possibility to filter data traffic caused by communication.

Unfortunately, itemised bills only contain outgoing connections, which allows a single-sided view only. Nevertheless, persons that communicate on a regular base with the user are placed closer to the user and result in a larger node than others. Therefore, the distance and the size of the nodes can be seen as a measure of strength of a relationship. The larger and closer a node to the centre is, the more interaction took place. This measure is calculated by counting the seconds of telephony plus the SMS count multiplied by 60.

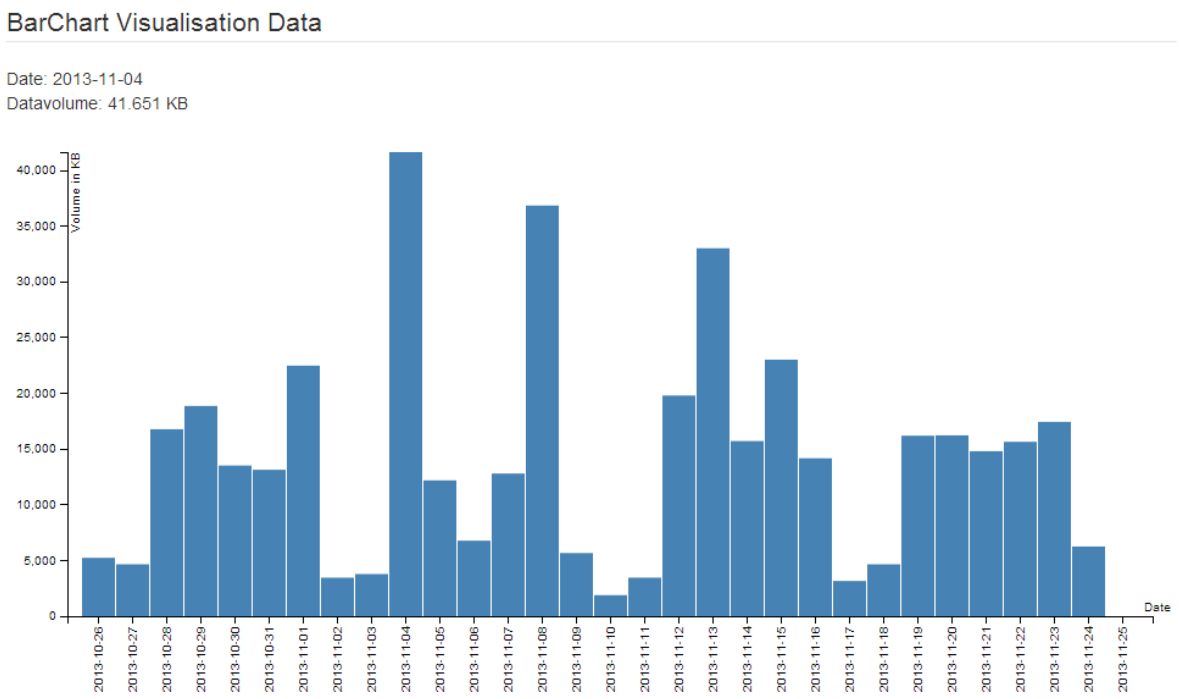


Figure 5.3: Screenshot of the bar-chart data visualisation

5.2.2 Scenario 2: Single-user and Multi-source

In the second scenario, we are investigating the social network of two or more sources of one user (e.g. itemised bills and the Whatsapp message history). The aim of the social network analysis is to discover persons, who are connected only with one means of communication (e.g. only via Whatsapp or only via TSP). Therefore, one itemised bill as well as the Whatsapp message history of the same user are provided to the application. While the bar-charts and stream-graphs are unclear (lots of dates on the x-axis), the force-graph allows detailed analysis with the help of the full-screen and sticky-node capabilities. Due to the large number of nodes, the full-screen option hides the file loading part on the left side of the application and allows the force-graph to occupy the whole browser width. Sticky-nodes describes the capability of a node to become fixed on the screen. If a node is dragged, it can be moved and stays fixed wherever it is dropped. Unlocking its position is possible by double-clicking it again. To relocate sticky-nodes on the screen, these nodes have a red border. While Whatsapp does not contain the users telephone number, the term *MY PHONE* is used as origin of the Whatsapp social network visualisation. The itemised bill creates another source with the real telephone number. Each source creates its own connected nodes. Only some nodes occur in both datasets and are linked to both sources giving the node more influence by combining their size measures. Figure 5.8 shows the results with Whatsapp as source on the left, itemised bills on the right and the overlapping nodes fixed in the middle.

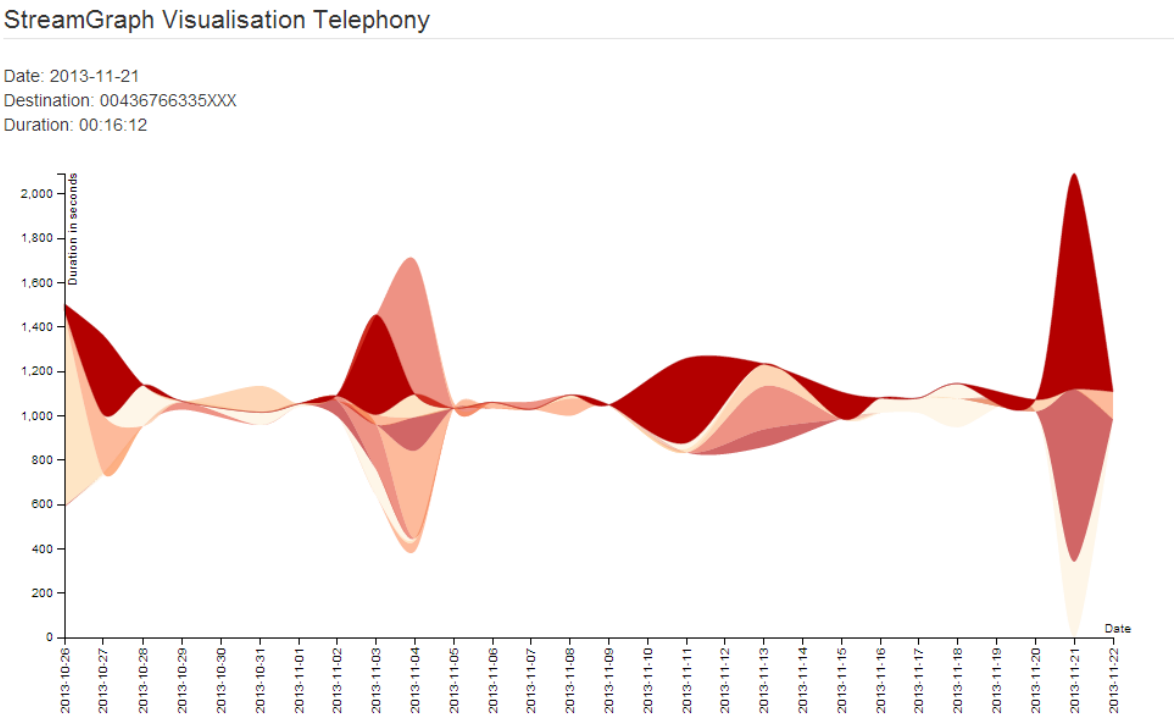


Figure 5.4: Screenshot of the stream-graph telephony visualisation

5.2.3 Scenario 3: Mulit-user and Multi-source

Another interesting outcome can be created by the combination of two itemised bills from different users who have an overlapping group of friends or colleagues. In this case, two source nodes are created, each with it's own connected nodes. Some of the nodes, like the social network of the users overlap. This is represented by nodes having a connection to both source nodes. Additionally, the force-graph reveals, that one user is calling the other user more often (node size depends on outgoing communication attempts). Figure 5.9 shows two friends (source nodes left and right). The size of the left node shows, that the friend on the right side calls the left one more often. Additionally, three persons could be identified, who are contacted by both parties.

It is even possible to use sources from more users to investigate more complex networks. The final investigation shows a combination of three sources from different providers containing communication information of three users (see figure 5.10). Again, the graph shows one node for each user and the network corresponding to the users. Even with a high number of nodes on the screen, it was easy to identify the overlapping nodes and fix these on the screen with the sticky-node ability to create a clearly arranged visualisation of the social network.

So the force-graph provides interesting insights into social networks of one or more users in a clearly arranged manner.

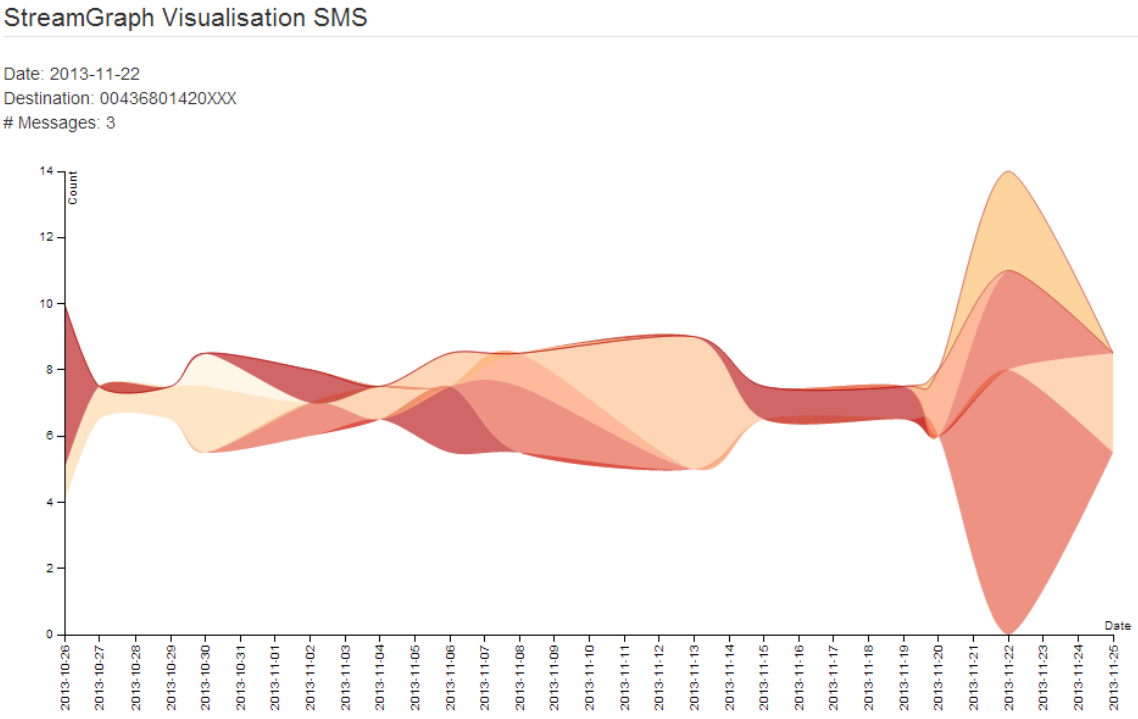


Figure 5.5: Screenshot of the stream-graph SMS visualisation

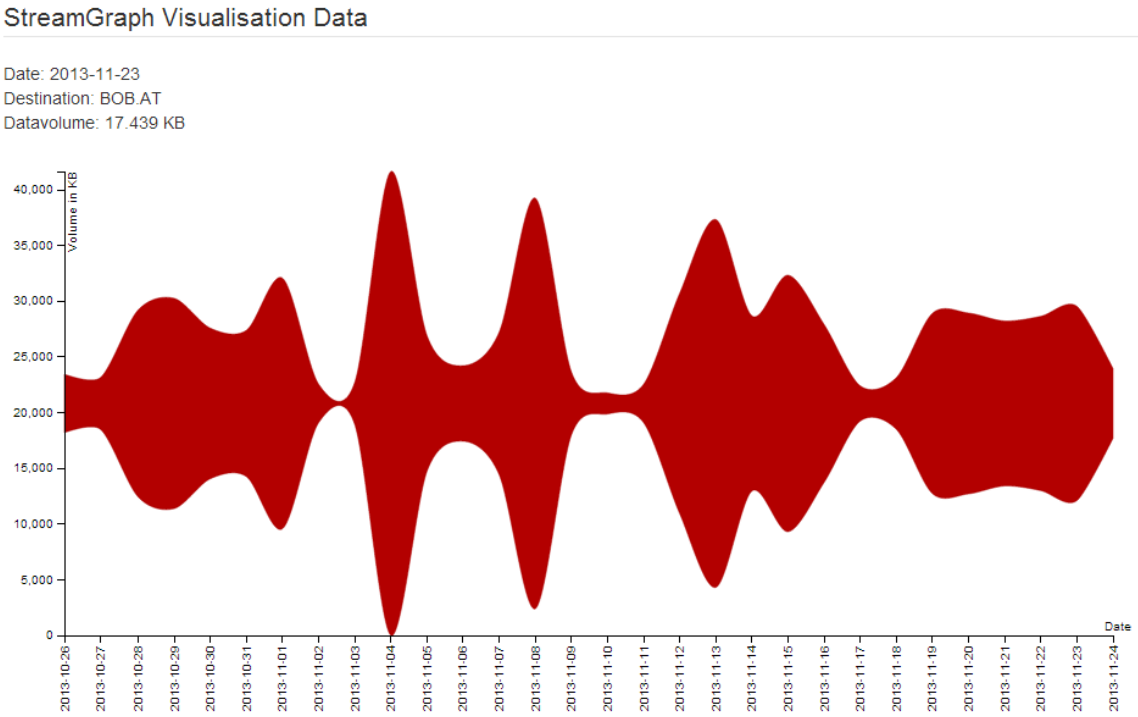


Figure 5.6: Screenshot of the stream-graph data visualisation

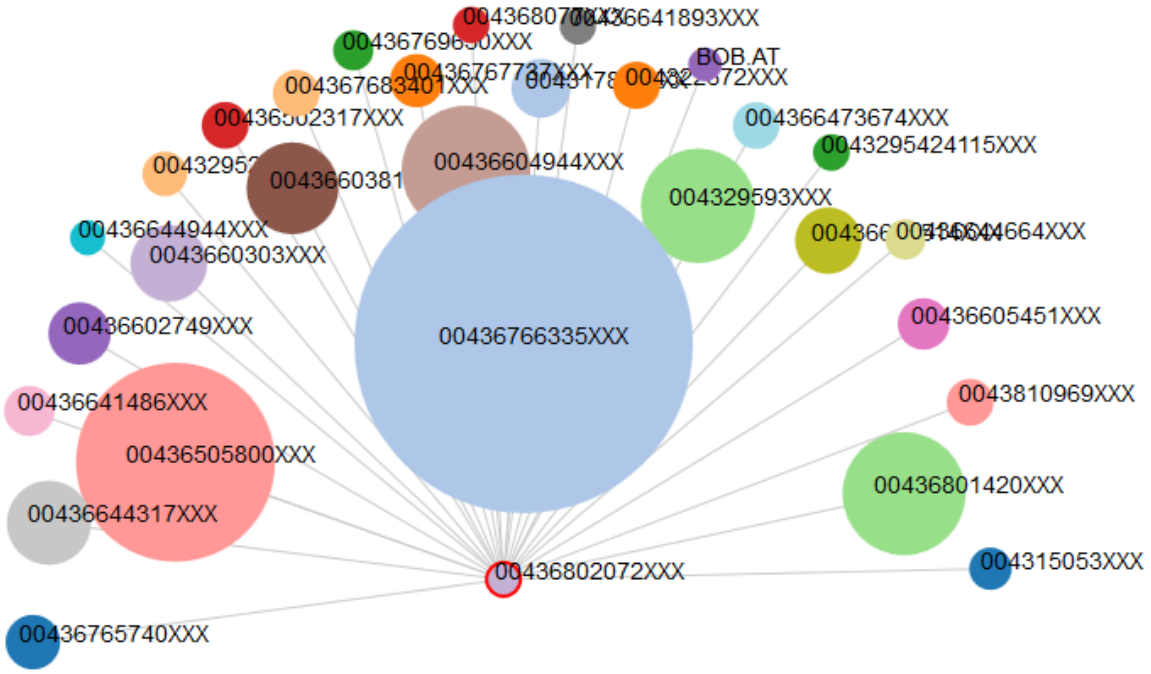


Figure 5.7: Screenshot of the force-graph visualisation

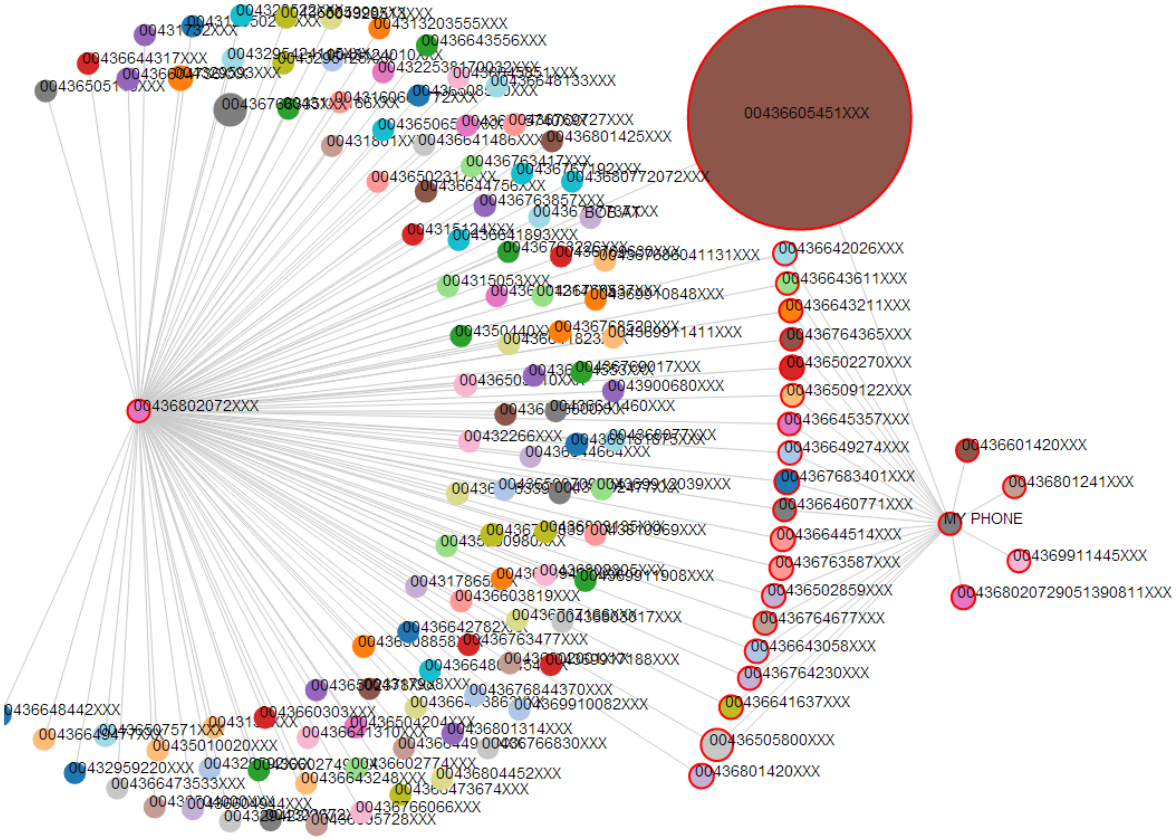


Figure 5.8: Screenshot of the force-graph visualisation combining one user's TSP itemised bill and the same user's Whatsapp message history

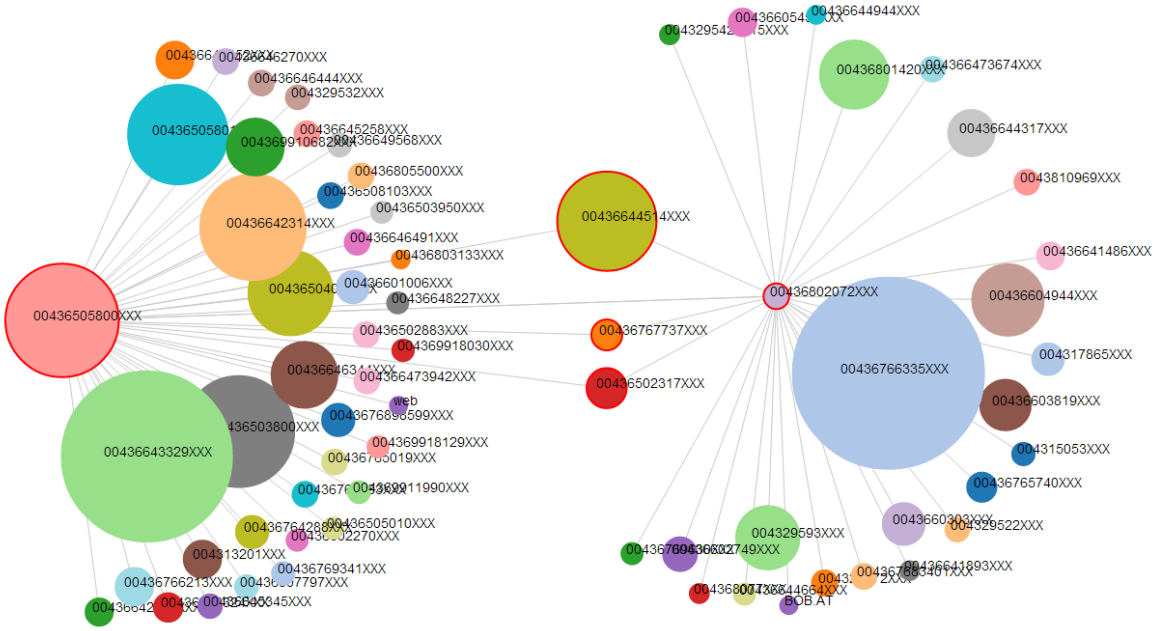


Figure 5.9: Screenshot of the force-graph visualisation combining itemised bills of two friendly users

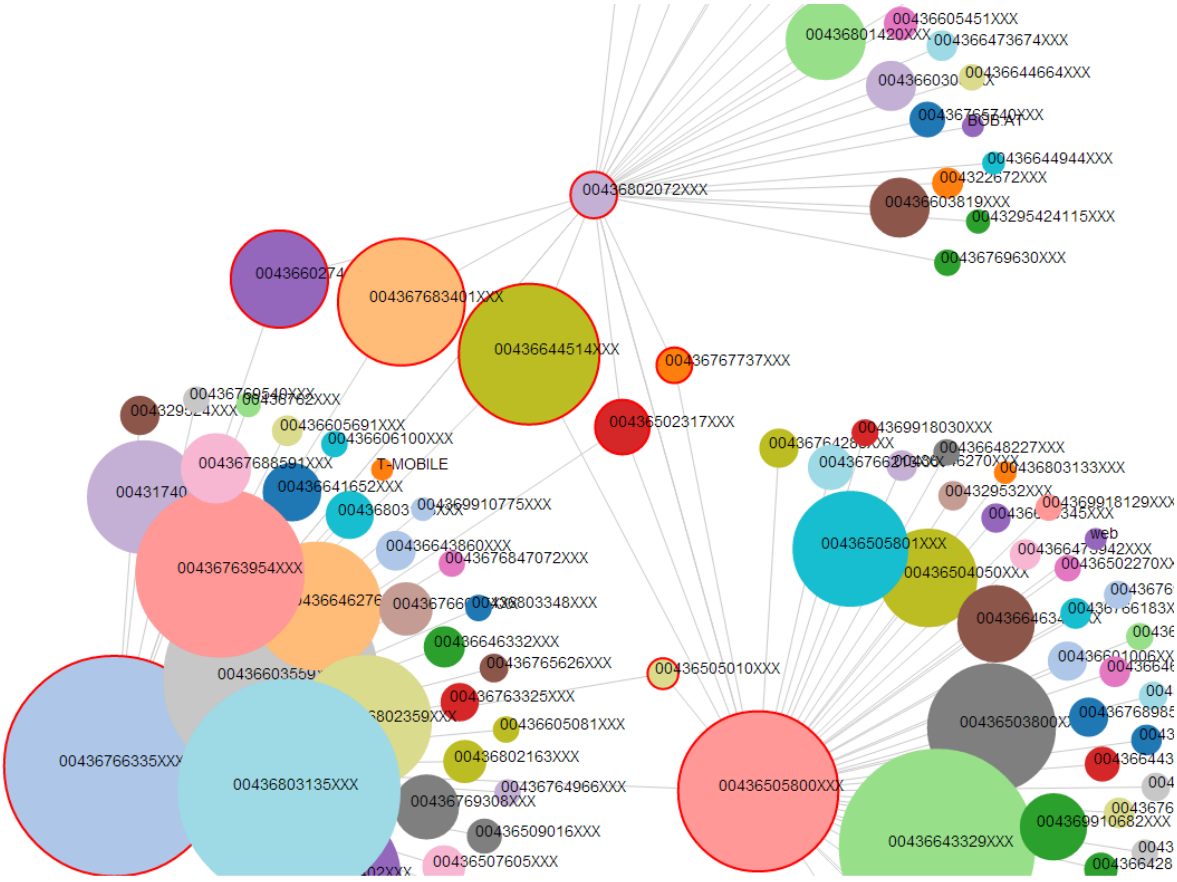


Figure 5.10: Screenshot of the force-graph visualisation combining itemised bills of three friendly users

Chapter 6

Conclusion and Future Work

Our work shows, that the visualisation of communication profiles provides improved readability in comparison to numbers on a itemised bill. Our application allows users to investigate communication patterns as well as the visualising their social network (see section 5.2). According to the bar-charts and the stream-graphs, reoccurring communication events can be identified and the social relations are visualised by the force-graph. In comparison to related work (see section 2.1), this project does not operate on a users email accounts or iPhone backup, but uses itemised bills to identify communication patterns. The client-side only implementation of our application provides maximum privacy to users, because it is not required to transmit the relevant information for analysis to a server. With the help of graphs, a user is now able to see the traces left behind in this digital world. Nevertheless, while itemised bills only contains outgoing connections, this visualisation is unfortunately incomplete. But, with the import feature for Whatsapp's message history, this problem could be solved at least for text messages.

In a future work, the visualisation component could be enhanced in a way to replace telephone numbers with names to improve the visual acceptance. Another step could provide the storage of the revealed communication patterns on the client computer for later investigation and further enrichment with user supplied data. Finally, bringing this application to a native mobile platform (e.g. Android or iOS mobile application) could simplify data acquirement by accessing messaging statistics directly on the phone.

Nevertheless, more and more communication takes place in secure or proprietary channels that cannot be accessed on the phone. Whatsapp, to name one representative messaging application, enciphers its communication database on the device in a way that no third party application has access to the stored messaging history. Actually, these databases can be decrypted, but the next step in application evolution could close this backdoor. So it is only a matter of time, before third party messaging applications can successfully prevent access to their messaging history and only the application provider can create statistics about one's messaging patterns and social network activities. As we have seen at recent initial public offerings (IPO) (e.g. Facebook) and take overs (e.g. Whatsapp taken over by Facebook), companies based on social networks and communication services, can earn a lot of money with our communication data.

Bibliography

[1] Radu Tyrsina. How nsa surveillance algorithms see into your life. http://techpp.com/2012/07/24/nsa-surveillance-algorithms-see-into-your-life/, 06 2013. iv, 8

[2] M. Newman. *Networks: An Introduction*. OUP Oxford, 2010. iv, 11, 12, 13, 14, 15, 16, 17, 18

[3] Stephen P. Borgatti, Ajay Mehra, Daniel J. Brass, and Giuseppe Labianca. Network analysis in the social sciences. *Science*, 323(5916):892–895, 2009. iv, 12, 13

[4] Smilkov Daniel, Jagdish Deepak, and Hidalgo César. Immerson. https://immersion.media.mit.edu/, 03 2014. a people-centric view of your email life. iv, 3, 19

[5] Bay Gross. Prismviz. http://prismviz.com/, 03 2014. your SMS history visualized. iv, 3, 20

[6] Mathew Bloch, Lee Byron, Shan Carter, and Amanda Cox. The ebb and flow of movies: Box office receipts 1986-2008. http://www.nytimes.com/interactive/2008/02/23/movies/20080223_REVENUE_GRAPHIC.html, 02 2008. iv, 21

[7] Lee Byron and Martin Wattenberg. Stacked graphs – geometry & aesthetics. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1245–1252, 2008. iv, 3, 21, 22

[8] Rtr telekom monitor 1/2014, 01 2014. iv, 23

[9] Andrew S. Tanenbaum and Maarten van Steen. *Distributed Systems: Principles and Paradigms*. Prentice Hall International, 2nd rev. ed. edition, 2008. iv, 26, 28

[10] S. Mavrody. *Sergey's Html5 & Css3 Quick Reference: Color Edition*. Belisso, 2010. iv, 32

[11] Black Duck Softare Inc. ohloh.net. http://www.ohloh.net/, 03 2014. Ohloh is a directory, a community, and analytics and search services. iv, 33

[12] M. Lesk. Big data, big brother, big money. *Security Privacy, IEEE*, 11(4):85–89, July 2013. vi, 10

[13] Adam Bard. adambard.com. http://adambard.com/blog/top-github-languages-for-2013-so-far/, 03 2014. Top github languages for 2013. vi, 31

[14] The European Parliament. Eu directive 2006/24/ec - data retention. http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2006:105:0054:0063:EN:PDF, 03 2006. 1, 4, 6

[15] The George Washington University. The national security agency declassified. http://www2.gwu.edu/~nsarchiv/NSAEBB/NSAEBB24/index.htm, 03 2005. Internet wiretapping mixes "protected" and targeted messages, Info Age requires rethinking 4th Amendment limits and policies. 1

[16] Ithiel de Sola Pool and Manfred Kochen. Contacts and influence. *Social networks*, 1(1):5–51, 1979. 2

[17] Glenn Greenwald and Ewen MacAskill. Nsa prism program taps in to user data of apple, google and others. http://www.theguardian.com/world/2013/jun/06/us-tech-giants-nsa-data, 06 2013. Top-secret Prism program claims direct access to servers of firms including Google, Apple and Facebook. 2

[18] United States Foreign Intelligence Surveillance Court. Verizon forced to hand over telephone data – full court order. http://www.theguardian.com/world/interactive/2013/jun/06/verizon-telephone-data-court-order, 06 2013. 2

[19] Court of Justice of the European Union. Press release no 54/14. http://curia.europa.eu/jcms/upload/docs/application/pdf/2014-04/cp140054en.pdf, 04 2014. 2

[20] Jonathan Mayer. Metaphone. http://metaphone.me/, 03 2014. A project to understand call and text privacy. 3

[21] Austrian Parliament. Federal law gazette i no. 27/2011 (bundesgesetzblatt. https://www.ris.bka.gv.at/Dokumente/BgblAuth/BGBLA_2011_I_27/BGBLA_2011_I_27.pdf, 05 2011. 4, 6

[22] Elaine Barker, William Barker, William Burr, William Polk, and Miles Smid. Nist special publication 800-57. *NIST Special Publication*, 800(57):1–142, 2007. 4

[23] Bruce Schneier. Bruce schneier on security. https://www.schneier.com/blog/archives/2013/09/the_nsas_crypto_1.html, 09 2013. The NSA's Cryptographic Capabilities. 4

[24] The European Parliament. Eu directive 1995/46/ec on the protection of individuals with regard to the processing of personal data and on the free movement of such data. http://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX:31995L0046, 10 1995. 5, 6

[25] F. Zuiderveen Borgesius. Behavioral targeting: A european legal perspective. *Security Privacy, IEEE*, 11(1):82–85, Jan 2013. 6

[26] United States Court. Olmstead v. united states. `http://www.law.cornell.edu/supremecourt/text/277/438`, 06 1928. 7

[27] United States Court. Katz v. united states. `http://www.law.cornell.edu/supremecourt/text/389/347`, 12 1967. 7

[28] Charlie Savage. N.s.a. chief says surveillance has stopped dozens of plots. `http://www.nytimes.com/2013/06/19/us/politics/nsa-chief-says-surveillance-has-stopped-dozens-of-plots.html`, 06 2013. 8

[29] S. Landau. Making sense from snowden: What's significant in the nsa surveillance revelations. *Security Privacy, IEEE*, 11(4):54–63, July 2013. 8

[30] Whitfield Diffie and Susan Landau. Communications surveillance: Privacy and security at risk. *Queue*, 7(8):10:10–10:15, September 2009. 8, 9

[31] Alan Rusbridger. The snowden leaks and the public. *The New York Review of Books*, 60(18), 11 2013. 9

[32] David Cole. We kill people based on metadata. The New York Review of Books - Blog, `http://www.nybooks.com/blogs/nyrblog/2014/may/10/we-kill-people-based-metadata/`, 05 2014. 9

[33] Nicole Perlroth, Jeff Larson, and Scott Shane. Secret documents reveal n.s.a. campaign against encryption. `http://www.nytimes.com/interactive/2013/09/05/us/documents-reveal-nsa-campaign-against-encryption.html`, 09 2013. 9

[34] Adrian Cho. Network science at center of surveillance dispute. *Science*, 340(6138):1272, 2013. 9

[35] IAB and IESG. Ietf policy on wiretapping. RFC 2804 (Informational), May 2000. 9

[36] S.M. Bellovin, M. Blaze, S. Clark, and S. Landau. Going bright: Wiretapping without weakening communications infrastructure. *Security Privacy, IEEE*, 11(1):62–72, Jan 2013. 9

[37] Jessica Leber. How wireless carriers are monetizing your movements. `http://www.technologyreview.com/news/513016/how-wireless-carriers-are-monetizing-your-movements/`, 04 2013. 10

[38] P. Kotzanikolaou. Data retention and privacy in electronic communications. *Security Privacy, IEEE*, 6(5):46–52, Sept 2008. 10

[39] C. Kadushin. *Understanding Social Networks: Theories, Concepts, and Findings*. Oxford University Press, USA, 2012. 11

[40] J. Scott and P.J. Carrington. *The SAGE Handbook of Social Network Analysis*. SAGE Publications, 2011. 11, 12, 13

[41] F. Harary, R.Z. Norman, and D. Cartwright. *Structural models: an introduction to the theory of directed graphs*. Wiley, 1965. 11

[42] E. O. Laumann, P. V. Marsden, and D. Prensky. *The boundary specification problem in network analysis.*, pages 18–34. Sage Publications, London, 1983. 12

[43] M. Berger and R.M. MacIver. *Freedom and control in modern society.* Van Nostrand series in sociology. Octagon Books, 1964. 13

[44] Jeffrey Heer, Nicholas Kong, and Maneesh Agrawala. Sizing the horizon: The effects of chart size and layering on the graphical perception of time series visualizations. In *In Proc. ACM Human Factors in Computing Systems (CHI*, pages 1303–1312, 2009. 22

[45] Gsm doc 28/85, services and facilities to be provided in the gsm system, rev. 2, 06 1985. 23

[46] Statistics Austria. Ict usage in households - usage of mobile broadband connectivity. online, 10 2013. 23

[47] Statistics Austria. Ict usage in households - usage for private purposes. online, 10 2013. 23

[48] Whatsapp Inc. Whatsapp. `http://whatsapp.com/`, 07 2014. cross-platform mobile messaging app. 24

[49] Sebastian Schrittwieser, Peter Fruehwirt, Peter Kieseberg, Manuel Leithner, Martin Mulazzani, Markus Huber, and Edgar R. Weippl. Guess who is texting you? evaluating the security of smartphone messaging applications. In *Network and Distributed System Security Symposium (NDSS 2012)*, 2 2012. 24

[50] Aditya Mahajan, M. S. Dahiya, and H. P. Sanghvi. Forensic analysis of instant messenger applications on android devices. *CoRR*, abs/1304.4915, 2013. 24

[51] E. F. Codd. A relational model of data for large shared data banks. *Commun. ACM*, 13(6):377–387, June 1970. 27

[52] Goggle Inc. angularjs. `https://angularjs.org/`, 03 2014. HTML enhanched for web apps. 32

[53] Bastian Mathieu. gephi. `https://gephi.org/`, 03 2014. Gephi is an interactive visualization and exploration platform for all kinds of networks and complex systems, dynamic and hierarchical graphs. 35

[54] Mike Bostock. d3.js. `http://d3js.org/`, 03 2014. data-driven documents. 35

[55] Michael Bostock, Vadim Ogievetsky, and Jeffrey Heer. D3: Data-driven documents. *IEEE Trans. Visualization & Comp. Graphics (Proc. InfoVis)*, 2011. 35