Technische Universität Berlin

Institut für Telekommunikationssysteme
Fachgebiet Architektur der Vermittlungsknoten

Fakultät IV
Franklinstrasse 28-29
10587 Berlin
http://www.av.tu-berlin.de

Diploma Thesis Proposal

Contextual and Conditional Content Distribution

Abdulrahman Hammood

Matriculation Number: 227675
01.01.2050

Supervised by
Prof. Dr. Thomas Magedanz

Assistant Supervisor
Dr. Adel Al-Hezmi

Contents

List of Figures

1 Introduction

The Internet has become an essential part of our daily lives in different sectors business, social communications, healthcare, etc. It has revolutionized our economy and society and being therefore considered at the top of the technological revolution in the current century. The success of the Internet can be seen on its traffic growth. The monthly global Internet traffic is expected to quadruple between 2010 and 2015 growing up from about 20.2 exabytes to 80.5 exabytes (one exabyte equals one billion gigabytes)[3]. Indeed, this growth indicates how huge the content is (and being increased rapidly) that is uploaded and consumed. Around one million minutes of video content will cross the network per second in 2015 [3]. Around one hour You-tube videos are being uploaded per second and more than four billion views per day[13]. The video-sharing content in You-tube is only one example of a huge number of distributed contents on the Internet provided by various content delivery platforms. These platforms provide different types of contents, i.e. contents are heterogeneous and can be anything, e.g. video, multimedia, books, etc.

The recent growth of multimedia content offered by multiple professional content providers (e.g. IPTV or mobile TV provider), available in several multimedia-based social networking communities or distributed in various user devices seems to be clear evidence for the need of an efficient multimedia provisioning framework that supports efficient and personalized provisioning and discovery mechanisms of multimedia content information comparing to the classical client-server provisioning systems. This thesis will address arise from the wealth of distributed multimedia content either in any controllable network or in user private network. The challenge is to provide users with technical means for rapid and instant access to relevant, trustworthy multimedia content information and enriched personalization.

Nowadays devices (e.g. PCs, smartphones, positioning devices, health monitors) in our environment are expected to work on high levels of independence, performing programmed actions that benefit their users in everyday life.In order to meet the set of requirements, these different devices are connected together performing certain tasks. This concept is known as Machine-to-Machine (M2M) communication. M2M is a concept that defines the rules and relations between devices while cooperating. It implies a highly automated usage of a set of devices simultaneously, without much need for human interaction. Although with the in-

crease of computational power, now it is even possible to run different M2M tasks on various consumer electronics (e.g. television sets, set-top boxes), smartphones are still more frequent used in M2M domain.

The aim of this thesis is to develop a context-aware platform in which context information of multimedia content environment (such as location and time) is considered and embedded into the captured multimedia content as content information or metadata. The proposed platform uses M2M concept, which provides several resources such as sensor informations like temperature, GPS data, and so on. These resources can be used to enrich the metadata of the captured content. The motivation behind this work is due to the lack of context/content aware storage management in the current Internet architecture which is one of its fundamental limitations [8]. Consumer, who is interested in certain multimedia content service, submits the request with defined conditions (e.g. time, location, content provider, etc.) that are evaluated against multimedia context information in order to deliver the associated multimedia content information (e.g. content resources, description, etc.). User-specific conditions can be published once or updated regularly. The multimedia content service shall examine user-specific conditions and notifies the user with matched multimedia content. Therefore, an efficient interactions model between consumer and the service will be developed. Furthermore, an end-to-end multimedia content service will be implemented in order to demonstrate the developed concept.

1.1 Motivation

Context-aware systems can help people in many areas of daily life to plan the daily schedule, to make important decisions correctly and perform other tasks instead of user.

Due to the fact that increased computing power of today's mobile devices, they perform tasks that were still a few years ago not possible. For these devices, with their increasingly complex applications, context-aware behaviour is of importance. Reliable and easy-used context-aware systems are required because of the explosive growth of content consumption from mobile and social interfaces and the consumer expectation of content availability. According to a statistic result reported by The Nielsen Company, U.S mobile video viewers have grown from 23 millions in the third quarter of 2010 to 31 millions in the third quarter of 2011.

M2M technology is growing fast and in the near future it will be available everywhere. Such a technology will help to enrich context information associated with multimedia content.

There are several reasons for working on this diploma thesis. On one hand, there is no correlation between contents and contexts nowadays supported by the

current Internet architecture. However, there is a demand for solutions or products that simplify the usage of the distributed content based on its contexts. The unavailability of such solutions or products is one of the main motivations behind this thesis. Within the context of this thesis, a new valid prototyping for context-aware content management platform will be developed. Therefore this thesis can set the ground for future investigation and can further be used as a cornerstone and give directions for design of better and generally accepted.

On the other hand, working on this thesis gives me the chance to get in-depth knowledge and hands-on experience of a hot topic that will evolve, improve, develop in the years to come and eventually will become inevitable part of normal way of living.

1.2 Objective

The main objective of this thesis is to develop a contextual content management platform in which the context information (metadata) is created automatically during content capturing and relying on local and distributed sensing information. The platform will support the correlation and synchronization of context information and multimedia content stream. To deliver the content to various devices the distribution mechanism will be implemented. The distribution mechanism has to be aware on device properties i.e screen resolution or internet speed. For context enrichment, the platform will use the benefits of M2M concept.

Figure 1.1 shows the main component of the platform:

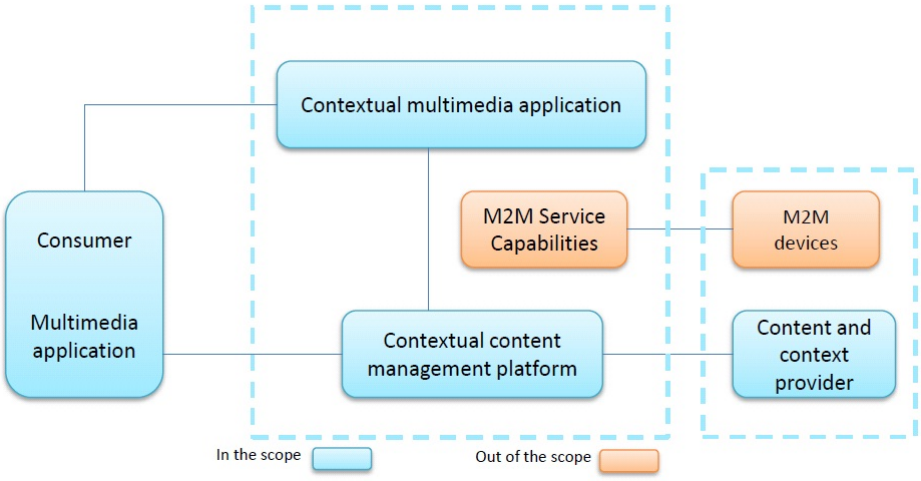

Figure 1.1: Overall Architecture

1.3 Scope

Due to lack of time and the possible wide range of technologies which have been mentioned above, the objective will be defined here in details to decide what should be developed in this thesis.

The scope of this thesis is to manage the relationship between content and context. For managing this relationship the thesis will study the state of the art of data model description and then choose the appreciate one. The thesis will also integrate M2M concept for context enrichment.

The activities in this thesis are outlined as follows:

- Act. 1: Study the state of the art of data model format, context description language
- Act. 2: Defining concept for discovering sensors (locally deployed or with M2M platform), subscriptions and notifications for sensor information.
- Act. 3: Investigate the process of automatic creation of context information and data fusion (correlation between context and content)
- Act. 4: Design the required management and delivery platforms
- Act. 5: Examine the available open sources for content management systems and HTTP-based streaming servers
- Act. 6: Based on available open source solutions, develop an end-to-end platform that enables content provider(mobile app.) to capture multimedia content with the associated context information and publish this content to the server and allows consumers(mobile app. or web based) to discover and subscribe to multimedia content according to defined conditions.
- Act. 7: Develop an end-to-end application to evaluate the implemented functionality
- Act. 8: Validate the implementation through an end-to-end deployment scenario that is planned to be deployed in the FOKUS open EPC(Evolved Packet Core) or MTC(Machine Type Communication) playground. In the first deployment scenario, the quality of content streaming to the consumer or network selection will be instantly identified according to user and network policies. In the second deployment scenario, content-related context information – in particular geographical and time information of multimedia content stream – can be considered as collaborative crowd sourcing with multimedia content that can enrich any MTC platform.

Figure 1.2 shows inputs and outputs for this thesis:

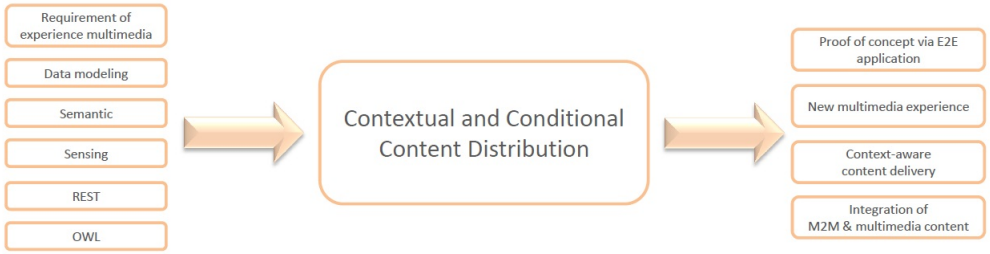

Figure 1.2: Inputs & Outputs

By studying these tasks the goal is to draw conclusions about best practices in this domain, and design the platform that can establish the basis for its further development and future implementation.

1.4 Methodologie

1.5 Outline

2 Background and Related Work

dkjekdede dejfekjfhkjehf jefkjefkjefkje jekfnkejfbkjebfkje kejfbnekjfbjkebfekjfnekjfbnkjebf kjenfk jbkejfkjenfkj nejk kjn kj kjbjkbkjbkjb kj bkj jk bkjb kjb kj bkj bkj bkj bkjb kj kjbkjbkjbkjbkj kjhkjhkjhjk kjhkjhjkh kjhkjhkjh kjhkjhk kjhjkhkj.

2.1 Context-awareness

In this section, the terms "context" and "context-aware computing" are discussed in more detail. Furthermore, application possibilities of context awareness is demonstrated.

In order for computers to assist users in their everyday tasks, they should adapt themselves to the current user's situation, and then respond according to this situation.

Humans are quite successful at conveying ideas to each other and reacting appropriately. This is due to many factors: the richness of the language they use, the common understanding of how the world works, and an implicit understanding of every-day situations. When humans talk with each other, they are able to use implicit situational information, or *context*, to increase the conversational bandwidth. Unfortunately, this ability to convey ideas does not transfer well to humans interacting with computers.VERGLEICHE mitDey and Abowd

2.1.1 What is Context?

The report that first introduces the term *context-aware*, [Schilit and Theimer] refers to context as location, identities of people and objects nearby, and changes to those objects. A similar definition, [Brown et al.] describes context as location, identities of the people around the user, the time of day, season, temperature, etc. [Ryan et al.] defines context as the user's location, environment, identity and time. [Dey] enumerates context as the user's emotional state, focus of attention, location and orientation, date and time, objects, and people in the user's environment. These definitions that define context by certain examples are difficult to apply. In order to determine whether certain types of information not listed in the definition are correct context or not, it is difficult to determine how we can use the definition to solve this dilemma.

A recent definition of context-awareness is described by [Dey and Abowd] who defines it as "Context is any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". This way it is easier for an application developer to enumerate the context for a given application scenario. If a piece of information can be used to characterize the situation of a participant in an interaction, then that information is regarded as context.

After the term "context" has been defined the next term "context-aware computing" is discussed.

2.1.2 Context-aware Computing

Context-aware computing was first described by [Schilit and Theimer] in 1994 to be software that "adapts according to its location of use, the collection of nearby people and objects, as well as changes to those objects over time". However, it is commonly agreed that context-aware computing was first investigated in 1992 by [Want et al.].

[Ryan] has also defined the term context-aware applications as applications that allow users to select from a range of physical and logical contexts according to their current interests or activities and also monitor input from environmental sensors.

2.1.3 M2M

2.1.4 Context Examples

The orientation of the screen of a tablet computer is automatically changed, maps can orientate themselves according to the user's direction with the zoom level adapted to the current speed, and the backlight of the phone is switched on when used in the dark.

These are examples of computers that are aware of their environment and their contextual use. However such functions were not common 10 years ago and only existed on prototype devices in research labs which researched context-aware computing.

Below are also some examples for context awareness in mobile and non-mobile environments. Although non-mobile environments for this thesis are not relevant, they are interesting at this point in order to show the diverse application areas which illustrate the usage Context-Awareness systems.

- identity

- spatial information
  e.g. location, orientation, speed, and acceleration

- temporal information
  e.g. time of the day, date, and season of the year

- environmental information
  e.g. temperature, air quality, and light or noise level

- social situation
  e.g. who you are with, and people nearby

- resources that are nearby
  e.g. accessible devices, and hosts

- availability of resources
  e.g. battery, display, network, and bandwidth

- physiological measurements
  e.g. blood pressure, heart rate,respiration rate,muscle activity, and tone of voice

- activity
  e.g. talking, reading, walking, and running

2.2 Context Description

In order to efficiently use the context data after acquisition, it needs to be represented and/or stored in an appropriate form suitable for further processing. Now some of the different types of context modeling will be discussed.

- **Key-value model:** This modeling technique represents contextual information with key-value pairs which is one of the most simple data structures for modeling contextual information. This model was already used by Schilit et al. [35] TODO to present the context by providing the value of a context information (e.g. location information) to an application as an environment variable. Distributed service frameworks frequently use the key-value modeling approach. Although key-value pairs lack capabilities for sophisticated structuring for enabling efficient context retrieval algorithms, they are easy to manage.

- **Logic based model:** This model is based on facts, expressions and rules. [10] A logic based system manipulates with the elemental items of this model and infers higher level logic by utilizing the already defined rules to deduce new facts.TODO cite "A Context Modeling Survey"

  A logic defines the conditions on which a concluding expression or fact may be derived (a process known as reasoning or inferencing) from a set of other expressions or facts. To describe these conditions in a set of rules a formal system is applied. In a logic based context model, the context is consequently defined as facts, expressions and rules. Usually contextual information is added to, updated in and deleted from a logic based system in terms of facts or inferred from the rules in the system respectively. Common to all logic based models is a high degree of formality.TODO cite "A Context Modeling Survey"

  In early 1993 McCarthy and his group at Stanford [29, 30] TODO researched one of the first logic based context modeling approaches and published it as a "Notes on formalizing contexts". They introduced contexts such as abstract mathematical entities with properties useful in artificial intelligence.

- **Ontology based model:** Ontologies are a promising instrument to specify concepts and interrelations [43, 20]TODO. The Web Onotlogy Language (OWL) is one way of implementing these ontologies. This consists of a set of classes, class hierarchies, set of property assertions, constraints on these elements, and types of permitted relationships between them. [12]TODO Another way to implement the ontologies is to use a knowledge representation language - the Resource Description Framework (RDF). This is a promising model because of the possibility to apply reasoning techniques. [8]

  Ötztürk and Aamodt [31] TODO proposed one of the first approaches of modeling the context with ontologies.

  Psychological studies on the difference between recall and recognition of several issues in combination with contextual information were analyzed by them. The necessity of normalizing and combining the knowledge from different domains was derived from this examination. A context model based on ontologies due to their strengths in the field of normalization and formality was proposed by them.

- **Graphical models:** Using the Unified Modeling Language (UML) is another way of representing context, as well as using an extension of the Object-Role Modeling (ORM) with context information. [10]

- **Object-oriented models:** Object-oriented design of context benefits from the common properties object-oriented programming, such as inheritance,

encapsulation, reuse, and polymorphism. An architecture exists that uses a class ContextObject, which is inherited by other context-specific classes which implement the common abstract methods, convert data streams to context objects and vice versa, and provide well known interfaces to access the context's logic.

- **Markup languages:** These models have hierarchical structure composed of tags and attributes. User Agent Profile (UAProf) and Composite Capabilities/Preference Profile (CC/PP) are some of the specifications that describe the capabilities of mobile devices and different user agents, enabling the content providers to produce and deliver content suitable for each request.

2.3 Content Adaptation

bla

Content adaptation can be defined as "the set of measures taken against a dynamically changing context, for the purpose of maintaining a user experience of the delivered content as close to that of the original content as possible".[Ding and Li] "Content adaptation has been widely acknowledged as one of the most important aspects for context-aware ubiquitous content delivery".

2.3.1 Audios/Videos Adaptation

bla

2.3.2 Documents Adaptation

bla

2.4 Content Discovery

2.5 Content Distribution

2.5.1 Stream

2.5.2 Download

2.6 Related Technologies

2.6.1 Web Services

There are many definitions for the term Web service, the World Wide Web Consortium (W3C) defines it as follows:[1]

> A Web service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards.

The W3C also states:[1]

> We can identify two major classes of Web services, REST-compliant Web services, in which the primary purpose of the service is to manipulate XML representations of Web resources using a uniform set of "stateless" operations; and arbitrary Web services, in which the service may expose an arbitrary set of operations.

By using Web Services, it is now very easy to make existing data and functions from existing applications available to consumers. Web services are considered as a "machine to machine" communication, which exchange messages via standard protocols.

The most known web services technologies TODO akronom SOAP and TODO REST are discussed in the following sections.

2.6.2 SOAP

SOAP stands for "Simple Object Access Protocol" and it relies on TODO akro. XML for its message format. For message negotiation and transmission, it is

dependent on other Application Layer protocols, particularly Hypertext Transfer Protocol (HTTP) or Simple Mail Transfer Protocol (SMTP), however HTTP has gained wider acceptance as it works well with today's Internet infrastructure and also with network firewalls.

A SOAP message is a type of envelope or container, which may contain an optional header element and a mandatory body element, see figure 2.1. Meta-data for this message are located in the header and the user data are stored in the body.

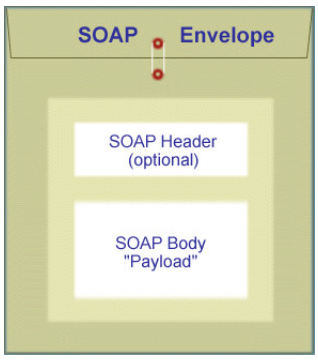

Figure 2.1: SOAP Envelop

SOAP Message Example: The following example gives an overview of a SOAP message:

```
<?xml version="1.0"?>
<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope">
  <soap:Header>
  </soap:Header>
  <soap:Body>
    <m:GetStockPrice xmlns:m="http://www.example.org/stock">
      <m:StockName>IBM</m:StockName>
    </m:GetStockPrice>
  </soap:Body>
</soap:Envelope>
```

2.6.3 REST

In addition to SOAP, there is another alternative for the implementation of Web services. Fielding in his dissertation describes an architectural style that he calls REpresentational State Transfer architecture or short REST.

REST is based on principles that are used in the largest distributed application - the World Wide Web. The World Wide Web is itself a gigantic REST application.

Many search engines, shops or booking systems have unintentionally been based on REST web services.

The REpresentational State Transfer Architecture is an architectural model, which describes how the Web should work. The model will serve as a guide and reference for future enhancements.

REST is not a product or standard. REST describes how web standards in a Web-friendly manner can be used.

REST Example: An online store will serve here as an example of a RESTful application. In this application, there are customers who can place items in shopping carts.

Each object of the application, such as the product or the customer is a resource that is externally accessible via a URL. With the following request in the example application, the shopping cart with the number 7621 is retrieved.

```
GET /cart/7621
```

It is not specified in REST how the result of a request is represented. Client and server must have a shared understanding how the data is represented, i.e. in XML or JSON TODO akro. The following example is a response in JSON format.

```
{
"customer": 7621,
"articles":[
        {"position":1,
        "articleNumber"=89,
        "description":"iPhone5",
        "price":200},
        {"position":2,
        "articleNumber"=76,
        "description":"Samsung Galaxy S III",
        "price":150}
]
}
```

2.6.4 SOAP vs. REST TODO

The main advantages of REST web services is that they are lightweight, without a lot of extra XML markup. Also REST has easy to read results and is easy to build requiring no special tool-kits.

SOAP also has some advantages, usually it is easy to use, provides relatively strong typing since it has a fixed set of supported data types, furthermore many

different kinds of development tools are available.

Next some aspects of SOAP and REST will be compared.

API Flexibility & Simplicity The key to the REST methodology is to use an interface that is already well known and widely used, the URI, in order to write web services. For example, providing a currency converter service, in which a user types-in the desired currencies for input and output and the specific amount in order to receive a real-time conversion, could be as simple as making a script accessible on a Web server via the following URI: http://www.currencyconverter.comconvert?in=us-dollar&value=100&out=euro

This service could easily be requested with an HTTP GET command by any client or server application with HTTP support. The resulting HTTP response depends on how the service provider wrote the script and it might be as simple as some standard headers and a text string containing the current price for the given currencies, or it might be an XML document.

The significant advantages of this interface method over SOAP-based services are as follows:-

The creation and modification of a URI in order to access different web resources can easily be figured out by any developer. However, in order for SOAP to be used, most developers would need a SOAP toolkit to form requests and obtain the results, as it requires specific knowledge of a new XML specification.

Bandwidth Usage The RESTful interface has short requests and responses, which is another advantage. Whereas, an XML wrapper around every request and response is required for SOAP. For a four- or five-digit stock quote, a SOAP response may require more than 10 times the number of bytes as the same response in REST, as SOAP requires namespaces and typing to be present.

Security The security perspective debate is probably the most interesting aspect of the comparison between REST and SOAP.

Sending remote procedure calls (RPC) through standard HTTP ports is seen by the SOAP camp as being a good way to ensure Web services support across organizational boundaries. In contrast however, REST followers see this as compromising network safety and considers this practice a major design flaw.

With REST, the administrator (or firewall) can discern the intent of each message by analyzing the HTTP command used in the request, even though the REST calls also go over HTTP or HTTPS. For example, a GET request is always seen as being safe because by definition, the data cannot be modified. It can only query data.

On the other hand, HTTP POST is used by a typical SOAP request to communicate with a given service. Without looking into the SOAP envelope, it is not possible to know whether the request simply wants to query data or delete entire tables from the database. This task is resource-consuming and it is not built into most firewalls.

On the downside with SOAP, the difficult task of authentication and authorization is left up to the application developer. However, the fact that the web servers already have support for these tasks, is taken into account by the REST methodology. REST methodology developers can make the network layer do all the heavy work by using industry-standard certificates and a common identity management system, such as an LDAP server.

However, REST is not perfect. It is not always the best solution for every web service. Data should never be sent as parameters in URIs in order to be kept secures.

Type Handling Due to its fixed set of supported data types, SOAP provides a stronger typing. In this way, it ensures a return value will be given in the corresponding native type in a specific platform. For example, when an API is HTTP based, the return value will need to be deserialized from its original XML format before being type-casted. However, handling complex data-types proves to be the main challenge and is mainly achieved by defining a serialization and deserialization mechanism, wherefore there is no definitive advantage concerning ease of client-side coding.

Client-side Complexity Making calls to an REST API poses less of a challenge than making calls to a SOAP API. While REST is elementary to all programming languages and merely implies constructing an HTTP request with the appropriate parameters, the latter requires a client library, a Stub and involves additional learning effort.

Testing and Troubleshooting A further characteristic of REST APIs is their easy testing and troubleshooting ability, requiring no more than a browser, the response appearing in the browser window itself. Generating a request does not require special test tools, this is a major advantage of REST based APIs.

Server-side Complexity The majority of programming languages provide easy to operate mechanisms to expose a method using SOAP. However exposing a method using REST based APIs, involves additional effort due to the task of mapping the URI path to specific handlers. Though various frameworks facilitate this task, the exposition of methods is still easier to achieve using SOAP than REST.

Caching To consume a REST based API service, a simple GET request is needed, therewith allowing proxy servers to cache their response very easily. In contrast, SOAP requests use POST and require a complex XML format, producing difficulties for response-caching.

2.6.5 NoSQL Databases TODO NoSQL Evaluation

SQL databases have been used to solve storage problems for a long time, including cases in which there is a high discrepancy between the object model and its relational model. The conversion of graphs to tables represents yet another disfunctional use of data mapping. The complex structure this sort of mismanagement causes depends on mapping frameworks and complex algorithms. The rigid relational scheme characteristic for SQL becomes especially inefficient for such web applications as blogs due to their multifaceted range of attributes that need to be stored in their respective tables, e.g. comments, pictures, audios, videos, source codes. Therefore adding or removing a new feature to this sort of website will necessarily result in system unavailability.

Nowadays of course, web sites are developing towards more interactive models, obliging databases to perform real-time scheme updates, thereby paving the way for NoSQL TODO akr to provide a database molded for modern demands.

There is a variety of ideas surrounding the NoSQL movement, however the core idea is to provide more flexible data models, as opposed to the SQL approach, in order to provide live scheme updates. The ever increasing amount of data streaming through the web implies challenges, which any competitive website wishing to stay in business will have to meet. Besides dealing with vast amounts of data, these sites have to respond to constant requests around the globe without allowing any noticeable latency.

To this end, many companies have developed their own storage systems, according to their specific needs, which have been classified as NoSQL databases. Considering the fact that these stores are set up to fulfill the individualized requirements of the companies they belong to, there can be no final answer as to which of them works most efficiently. For example, Facebook implemented the NoSQL database Cassandra in order to solve the so called "Inbox Search Problem" - the challenge of allowing Facebook users to search through their sent and

received messages - caused by the multitude of stored data alongside the high number of active users.

The following paragraphs will examine the structure and flexibility of four different data models offered by NoSQL systems, each accompanied by one or more exemplary implementations.

Key Value Stores: Key value stores are similar to maps or dictionaries where data is addressed by a unique key.

Key value stores are useful for simple operations, which are based on key attributes only. In order to speed up a user specific rendered webpage, parts of this page can be calculated before and served quickly and easily out of the store by user IDs when needed. Since most key value stores hold their dataset in memory, they are oftentimes used for caching of more time intensive SQL queries. Key value stores considered in this paper are Project Voldemort [6], Redis [7] and Membase [8].

Document Stores: Document Stores encapsulate key value pairs in ISDN or ISDN like documents. Within documents, keys have to be unique. Every document contains a special key "ID", which is also unique within a collection of documents and therefore identifies a document explicitly. In contrast to key value stores, values are not opaque to the system and can be queried as well. Therefore, complex data structures like nested objects can be handled more conveniently. Storing data in interpretable ISDN documents have the additional advantage of supporting data types, which makes document stores very developer-friendly. Similar to key value stores, document stores do not have any schema restrictions. Storing new documents containing any kind of attributes can as easily be done as adding new attributes to existing documents at runtime.

Document stores offer multi attribute lookups on records which may have complete different kinds of key value pairs. Therefore, these systems are very convenient in data integration and schema migration tasks. Most popular use cases are real time analytics, logging and the storage layer of small and flexible websites like blogs. The most prominent document stores are CouchDB [9], MongoDB [10] and Riak [11].

Column Stores: Column Family Stores are also known as column oriented stores, extensible record stores and wide columnar stores. All stores are inspired by Googles Bigtable [12], which is a "distributed storage system for managing structured data that is designed to scale to a very large size" [12]. Bigtable is used in many Google projects varying in requirements of high throughput and latency-sensitive data serving. The data model is described as "sparse, distributed, persis-

tent multidimensional sorted map" [12]. In this map, an arbitrary number of key value pairs can be stored within rows. Since values cannot be interpreted by the system, relationships between datasets and any other data types than strings are not supported natively. Similar to key value stores, these additional features have to be implemented in the application logic. Multiple versions of a value are stored in a chronological order to support versioning on the one hand and achieving better performance and consistency on the other one (chapter four). Columns can be grouped to column families, which is especially important for data organization and partitioning (chapter five). Columns and rows can be added very flexibly at runtime but column families have to be predefined oftentimes, which leads to less flexibility than key value stores and document stores offer. HBase [13] and Hypertable [14] are open source implementations of Bigtable, whereas Cassandra [15] differs from the data model described above, since another dimension called super columns is added. These super columns can contain multiple columns and can be stored within column families. Therefore Cassandra is more suitable to handle complex and expressive data structures.

Graph Stores:

2.6.6 Application Messaging

2.6.7 Search Engines

Apache Solr

Elasticsearch

2.6.8 Web Servers

Apache

NginX

2.6.9 Spring Framework

2.6.10 Application Servers

2.7 Related Work

3 Requirements

As described in section 1.2, the objective of this thesis is to design and develop a generic contextual content management platform, which supports the correlation of context information and multimedia content, content discovery and content distribution. In the following sections, the problem statement is discussed furthermore providing two instructive use cases. The use cases illustrates the functional assets of the framework and facilitate a deduction of its non-functional requirements.

3.1 Problem Statement

The evident growth of multimedia content offered by highly competitive providers alongside the technological progress concerning portable internet-ready devices(e.g. iPhone, iPad) calls for a more efficient correlation between content requests on the one hand and personalized search results and discovery mechanisms provided by the framework on the other. The challenge is to provide users with technical means for rapid and instant access to relevant, trustworthy multimedia content information and enriched personalization.

TODO

3.2 Scenarios

For better comprehension of the functionality of the proposed platform, two examples are given in this section as use case scenarios.

Mobile capturing of live event A user - as a content provider - captures a live event (e.g. demonstration, car race, marathon, Tour de France, etc.) using an application on a GPS capable smart phone. While filming, the application also collects context information (e.g. location, acceleration, temperature, time, etc.). Later the user uploads the captured content with its context information to the platform.

Let's consider a video of the Tour de France as an example for the uploaded content. Any consumer, who is interested in a specific uploaded video that has been taken in a specific place on the road of the tour (e.g. Les Essarts: town

which is located in western France), searches for the video by specifying some related information such as time range, location and "Tour de France" as search string. The platform will then give the user a list of all videos that match the specified criteria. The user can select any of the listed videos and begin streaming.

Restaurant guide Another approach would be a domain oriented search (e.g. restaurants, gas-station's, public libraries...). Considering a hungry user looking for a suitable restaurant; the restaurant guide will help users find facilities based on search criteria e.g name, place, cuisine, ratings ... etc. and then display images or videos of the selected restaurant and other information e.g price list, daily menu etc.

3.3 Functional requirements

In 'normal' applications, the functional requirements serve the purpose of describing what the system should do. This applies to both internal processes, and the interaction of the system with its environment. These requirements are derived commonly from use cases. Frameworks are different, identifying functional requirements is more difficult. Frameworks usually do not address specific use cases but are supposed to be open for varied scenarios. The functional requirements of the framework are therefore rather abstract.

R1 User Management Administrators of the framework can grant access to other administrators or providers, thereby ensuring a simple user management. Once a provider has been registered, he can administrate the flow and accessibility of multimedia content via his respective applications.

Applications Management The framework should provide an easy mechanism for creating and deleting applications. When a provider wishes to create a new application he sends a request including global configurations. A provider may also grant access to other providers to use his applications. This access either grants full administrative control over the applications or can be limited to uploading/downloading and searching activities.

Content & Context Data Store The framework should furthermore provide a mechanism for creating data stores for multimedia contents and their related context. In order to ensure legitimate context-based search results, the framework facilitates the correlation between content and its related context.

Content Discovery In order for the provider to discover context-based contents, an adjustable search engine is required. To this end, the provider defines the parameters relevant to potential search requests. The provider can respond to new challenges and refine search options as time goes on by adding new parameters to the original set.

Content Adaptation Based on the global configuration of the application, the framework should support content transcoding. Transcoding refers to optimizing processes, e.g. quality adjustments for efficient use in varying networks (mobile, Wifi), size adjustment for individual displays in the area of video contents or Word to PDF conversions for textual contents.

Content Distribution The framework should support content delivery to most widespread internet-able devices. It should also optimize delivery by streaming contents from the nearest available server, thereby minimizing the network latency and reducing bandwidth costs.

3.4 Non functional requirements

Non-functional requirements are mainly related to quality aspects of a system. As a implementation of the design presented in Chapter 4, which is considered in this work as a prototype, the non-functional requirements play a subordinate role. However, some non-functional requirements have quite an influence on fundamental architectural decisions. So it is nevertheless important to analyze these requirements. The following section outlines the non-functional requirements for the development of the framework.

3.4.1 Usability

Since this thesis concerns the development of a framework and not a concrete application or GUI this quality feature's applicability is limited. However there are certain ease-of-use requirements constituting the degree of effort needed to comprehended, evaluate and effectively use the software that are relevant.

All functionalities of the framework shall be accessible in a simple way. From a developer's point of view, who in a sense represent the "user" of the framework, it can be said that a good structure and readability of the source code is desirable. Changes and enhancements to the framework shall always be restricted to as few logical components as possible. More importantly, however, the standard use of the framework shall not require deep knowledge of the internal structure of the framework's components. Creating a new application for instance, shall be

possible without further knowledge of the framework via the external interface of the framework.

3.4.2 Efficiency

The efficiency describes the response time for inquiries, as well as the consumption of resources. The framework shall be capable to serve multiple applications simultaneously. The creation of an additional application shall affect the performance of the overall system only marginally.

3.4.3 Changeability

3.4.4 Scalability

4 Design

4.1 Architecture Overview

4.2 Framework Components

4.2.1 Repository

4.2.2 Search Engine

4.2.3 Content Adaptation

4.2.4 Content Distribution

4.2.5 Application Messaging

4.2.6 User API

4.3 Interfaces

4.4 Requirement Fulfillment

5 Implementation

5.1 Development Environment

5.1.1 Eclipse

5.1.2 Maven

5.2 Framework Implementation

5.2.1 Spring Framework

5.2.2 Repository

MongoDB

5.2.3 Search Engine

ElasticSearch

5.2.4 Content Adaptation

FFmpeg

5.2.5 Content Distribution

HTTP-Live-Video-Stream-Segmenter-and-Distributor

NginX

5.2.6 Application Messaging

RabbitMQ

5.2.7 User API

SPRING DATA - REST

5.3 Components Integration and Configuration

5.4 REST API

6 Evaluation

6.1 Test Environment

6.2 Test Scenarios

6.2.1 Usability

6.2.2 Performance

7 Conclusion

Bibliography

[1] Web services architecture. URL `http://www.w3.org/TR/ws-arch/`.

[2] P.J. Brown, J.D. Bovey, and Xian Chen. Context-aware applications: from the laboratory to the marketplace. *Personal Communications, IEEE*, 4(5):58 –64, oct 1997. ISSN 1070-9916. doi: 10.1109/98.626984.

[3] Cisco. Cisco visual networking index: Forecast and methodology, 2010-2015. http://www.cisco.com/en/US/solutions/collateral/ns341/ns525/ns537/ns705/ns827/white_paper_c11-481360.pdf.

[4] Anind K. Dey. Context-aware computing: The cyberdesk project. In *AAAI 1998 Spring Symposium on Intelligent Environments*, pages 51–54, Palo Alto, 1998. AAAI Press. URL `http://www.cc.gatech.edu/fce/cyberdesk/pubs/AAAI98/AAAI98.html`.

[5] Anind K. Dey and Gregory D. Abowd. Towards a better understanding of context and context-awareness. In *Workshop on The What, Who, Where, When, and How of Context-Awareness (CHI 2000)*, The Hague, The Netherlands, April 2000. URL `http://www.cc.gatech.edu/fce/contexttoolkit/`.

[6] Jie Ding and Ning Li. A distributed adaptation management framework in content delivery networks. In *Wireless Communications, Networking and Mobile Computing (WiCOM), 2011 7th International Conference on*, pages 1 –4, sept. 2011. doi: 10.1109/wicom.2011.6040622.

[7] Roy Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, University of California, Irvine, 2000.

[8] EC FIArch Group. Fundamental limitations of current internet and the path to future internet. http://ec.europa.eu/information_society/activities/foi/docs/current_internet_limitations_v9.pdf.

[9] N Ryan. Mobile computing in a fieldwork environment: Metadata elements. Project working document, version 0.2, 19997.

[10] N. Ryan, J. Pascoe, and D. Morse. Enhanced reality fieldwork: the context-aware archaeological assistant. In V. Gaffney, M. van Leusen, and S. Exxon, editors, *Computer Applications and Quantitative Methods in Archaeology (CAA 97), Oxford*, 1997. URL `http://www.cs.ukc.ac.uk/research/infosys/mobicomp/Fieldwork/Papers/CAA97/ERFldwk.html`.

[11] B.N. Schilit and M.M. Theimer. Disseminating active map information to mobile hosts. *Network, IEEE*, 8(5):22 –32, sept.-oct. 1994. ISSN 0890-8044. doi: 10.1109/65.313011.

[12] Roy Want, Andy Hopper, Veronica Falcao, and Jonathan Gibbons. The active badge location system. *ACM Trans. Inf. Syst.*, 10(1):91–102, 1992. URL `http://dblp.uni-trier.de/db/journals/tois/tois10.html#WantHFG92`.

[13] Youtube. press statistics. http://www.youtube.com/t/press_statistics.