Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies

FIIT-13428-64550

Bc. Richard Roštecký

Interconnection of content delivery networks

Master's Thesis

Supervisor: Ing. Roman Broniš

May 2014

Slovak University of Technology in Bratislava
Faculty of Informatics and Information Technologies

Bc. Richard Roštecký

Interconnection of content delivery networks

Master's Thesis

Study program: Počítačové a komunikačné systémy a siete
Field of Study: 9.2.4 Computer Engineering
Place: Institute of Computer Systems and Networks,FIIT STU, Bratislava
Vedúci práce: Ing. Roman Broniš

May 2014

Anotácia

Slovenská technická univerzita v Bratislave
FAKULTA INFORMATIKY A INFORMAČNÝCH TECHNOLÓGIÍ
Študijný program: Počítačové a komunikačné systémy a siete
Autor: Bc. Richard Roštecký
Diplomová práca: Prepojenie sieti pre doručovanie obsahu
Vedúci práce: Ing. Roman Broniš
Máj 2014

Projekt sa zameriava na skvalitnenie doručovania obsahu vytvorením adaptéra na prepojenie CDN sietí (Content Delivery Network - CDN).

Teoretická časť práce sa zameriava na analýzu možností používateľských zariadení, ktoré dokážu používateľovi sprístupniť multimediálny obsah. Tiež sa zameriava na siete pre doručovanie obsahu, ktoré doručujú obsah na základe zemepisnej polohy a dokážu používateľovi zaistiť kvalitu doručenia obsahu. Hlavný dôraz je pri tom kladený na vysvetlenie problematiky a koncepcie sietí na doručovanie obsahu.

Praktická časť práce sa zameriava čiastocne návrhom a tvorbou služby pre používateľa, ale hlavná časť je venovaná návrhu a tvorbe adaptéra na prepojenie sietí pre doručovanie obsahu. Adaptér pozostáva z dvoch rozhraní. Tieto rozhrania sú webové servery. Rozhranie je implementované ako webová aplikácia. Komunikácia medzi rozhraniami prebieha prostredníctvom webových služieb. Rozhranie má za úlohu ovládať CDN sieť a administrátovi prostredníctvom webového rozhrania umožniť jeho nastavovanie.

Výsledkom práce je systém, ktorý dokáže prepojiť dve siete na doručovanie obsahu, ktorého súčasťou je administračné rozhranie.

Annotation

Slovak University of Technology Bratislava
FACULTY OF INFORMATICS AND INFORMATION TECHNOLOGIES
Degree Course: Computer and communication systems and networks
Author: Bc. Richard Roštecký
Master thesis: Interconnection of content delivery networks
Supervisor: Ing. Roman Broniš
May 2014

The project is focused on increasing quality of content delivery used by creating adapter to interconnection of content delivery networks (CDN).

The theoretical part of the thesis focuses on the analysis of the possibilities of user devices that will be handle multimedia content. Also it is focused on content delivery networks that deliver content based on geographical location to users and it can guarantee quality of delivery content. In theoretical part is the main emphasis to describe problems and concept of content delivery networks.

Practical part of the thesis focuses on the design and production service for the users that will be delivering multimedia content and main part is focused on design and implementation of adapter that interconnect content delivery networks. Adapter consists of two interfaces. These interfaces are web servers. The interface is implemented as web application. Communication between interfaces is based on web services. The purpose of interface is to control CDN and offer to administrator user interface to set it up.

Results of this work will be a system that allows interconnection of two content delivery networks with user interface for administrators.

DECLARATION

I declare that I elaborated this project by myself using referred literature.

.....

Bc. Richard Roštecký

Contents

List of Figures

List of Examples

1 Introducion

1.1 Introduction to the topic

Sharing and watching of multimedia content belongs to between the most recent activities of users on internet. Every user wish to have requested content downloaded as soon as possible in yours devices. When lot of people want access to the same content in same time then server where is content store can be deny of service. Especially when some sport event has live stream on internet then lot of user want to access on it. Therefore exist content delivery networks. Content delivery network is a set of servers that expand content from origin server to world and it grow up performance of origin server. Then users can get requested content in short time. The load on origin server is divided to servers belongs to content delivery network. Our effort in this project will be increase performance of delivery content based on interconnect more content delivery networks.

1.2 The focus and objectives of the project

In our work we will primarily focus on multimedia content and its delivery to end users.

At analysis we focus on properties of Samsung Smart TV, Android platforms and Content Delivery Networks. We will compare advantages and disadvantages of Android platform and Smart TV platform. It is important for next development. We focus on supported multimedia file formats for both platforms. Section about CDN will be focused on CDN structure, explain roles of part of CDN network and possibilities of interconnection between CDNs.

Origin goal of practical part of the project was create a system which will offer a service for users with movie library. Movie library will offer to watch movie at our devices and guaranteed delivery multimedia content to user. New goal of practical part of the project is create adapter to interconnect content delivery networks what increase performance of delivering content to end users.

1.3 Document overview

Document is separated into a few main parts. At beginning of document we can find introduction to the topic. Next section is Analysis with theoretical principles and structures of the

project. For this time, last part of the document is System design for delivery of multimedia content.

1.4 Used abbreviations

Table 1: *Table of abbreviations.*

AAC	Advance Audio Coding
API	Application Programing Interface
CDN	Content Delivery Network
CDS	Content Delivery System
CDSM	Content Delivery System manager
CSS	Cascading style sheets
CZ	Coverage Zone
DASH	Dynamic and Adaptive Streaming over HTTP
DI	Distribution Infrastructure
DNS	Domain Name System
DRM	Digital Rights Management
FTP	File Transfer Protocol
GUI	Graphical User Interface
HAS	HTTP Adaptive Streaming
HDTV	High Definition television
HLS	HTTP Live Streaming
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IIS	Internet Information Service
IP	Internet Protocol
ISO	International Organization for Standardization
ISP	Internet Service Provider
MPD	Media Presentation Description
MPEG	Movie Picture Expert Group
MPEG-TS	MPEG Transport Stream
OIPF HAS	Open IPTV Forum HTTP Adaptive Streaming
POPs	Points of Present

QoS	Quality of Services
RFQDN	Router Fully Qualified Domain Name
RRI	Request-Routing Infrastructure
RRPS	Request Routing Peering system
RTP	Real Time Protocol
RTSP	Real Time Streaming Protocol
SDK	Software Development Kit
SE	Service Engine
SR	Service Router
SS	Surrogate Server
SSO	Single Sign-On
TV	Television
UI	User interface
URL	Uniform Resource Locator
XML	Extensible Markup Language
3GPP	The 3rd Generation Partnership Project

1.5 Time plan

1.5.1 Diploma project 1

1. week: Analysis of CDN
2. week: Analysis of CDN
3. week: Analysis of CDN
4. week: Analysis of CDN
5. week: Analysis of Samsung Smart TV
6. week: Analysis of Android platform
7. week: Analysis of Android platform
8. week: Analysis of Android platform
9. week: Analysis of existing solutions
10. week: Analysis of existing solutions
11. week: System design for delivery of multimedia content
12. week: System design for delivery of multimedia content
13. week: System design for delivery of multimedia content

1.5.2 Diploma project 2

1. week: Study development of Android apps
2. week: Start creating of Android app
3. week: Study software for creating HLS streams
4. week: Creating HLS streams
5. week: Creating and testing variable HLS stream
6. week: Setting up CDN to delivery content
7. week: Study CDN interconnection
8. week: Study CDN interconnection
9. week: Desing CDN interconnection
10. week: Desing CDN interconnection
11. week: Creating document
12. week: Creating document

1.5.3 Diploma project 3

1. week: Implementation CDNi management
2. week: Implementation CDNi interface
3. week: Implementation CDNi interface
4. week: Implementation CDNi interface
5. week: Implementation Mobile application
6. week: Implementation Mobile application
7. week: Debuging and Testing system
8. week: Debuging and Testing system
9. week: Finalize document

2 Analysis

We divided the analysis of this theme into four main parts. First will be Samsung Smart TV. At context about Samsung Smart TV we will want to know how application works, which technologies are used for development and which options are provided by Smart TV with multimedia content. Our work will contain multimedia content therefor we will need to know which multimedia content is supported by Samsung Smart TV. Statistics of using Smart TV are important for future development and it will have a special part of analysis. Second part will be focused on content delivery network. We will try to answer questions on CDN function, what is structure of CDN and what are main parts of CDN. CDN will be important part of our system. Third part of analysis will be about Android platform. This platform we added to analysis during work because time brought more information about Smart TV and no good concepts of Smart TV. The last part of analysis will be short view of existing systems which are similar to our.

2.1 Samsung SMART TV

2.1.1 Overview

Samsung SMART TV introduces different view of using of televisions. Samsung SMART TV supports functions such as SMART Interaction, SMART Content and SMART Evolution. SMART Interaction provides control of television by motion control. Person can say "Turn off" and television turns off. SMART Content provides access to information by set of applications. SMART Evolution is a feature that supports simple upgrade of functions of television. These features ensure that the user doesn't need to buy new television when there are new features available. [1]

Functions of SMART TV are dependent to connection to Internet. Smart TV is a web based application running on an application engine installed on digital TVs. Applications are special web pages that are implemented on a web technologies and run on a TV screen. Viewing an application on the TV is similar to viewing web pages using a web browser. The main difference is in the user interface. User interface for SMART TV is the remote control.

2.1.2 SMART TV Application

Samsung SMART TV allows developers to create TV-oriented applications with extended functions by plugins. Unlike general web pages, application for Samsung SMART TV allows users to take advantage of TV-specific features. That can be application plugins which can allow users to change the application's volume or play a video that is not the part of the television broadcast, but it can be functions, which are provided by the File API to allow users to use a file storage systems. [1]

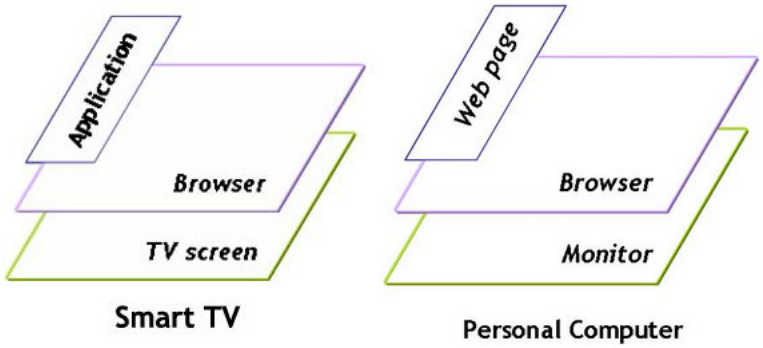

Figure 1: *Compare Smart TV and Personal Computer – Structure [1]*

Technically, a SMART TV application is a web page consists of HTML page, CSS file, JavaScript file, Config.xml file and optional Flash files. HTML page file creates the application's structure. SMART TV supports HTML 5. CSS file defines styles. JavaScript file controls application behavior. Config.xml file contains information about the operating environment and the application's version. In time of writing this analysis, SMART TV used WebKit application engine. [1]

Samsung SMART TV applications run on a browser. SMART TV also includes an Application Manager that handles applications-related tasks.

Application Manager is a part of Samsung SMART TV, that takes care about application in television. Application Manager can install, run and delete applications, also provides the common modules (e.g.: widget object, plugin object) that enables applications to run normally on the television screen. Also recognizes and identifies remote control button events. One part of Application Manager is an accounts manager that provides Single Sign-On (SSO). Single Sign-On requests enter login information of user only once. That means, user doesn't need to

enter account information for every access to application by remote controller. SSO Common module provides encryption and user accounts storage in SMART TV local storage. [1]

2.1.3 Project Types

During developing Samsung SMART TV applications, we can choose one of two types of project at SMART TV SDK. First type of project is JavaScript project. JavaScript API provides low-level platform functions that give you greater control over applications tasks and processes. Second type of project is Flash project. Flash project supports standard flash functionality. [1]

2.1.4 Streaming video to device

The main goal of our diploma thesis is to create a media service, that will deliver content with quality of services to end users. Samsung Smart TV provides several options for playing streamed video. For as much as Smart TV application supports HTML 5, there is the possibility of use video tag. [1]

HTML5

HTML 5 defines an element, video tag, that specifies a standard way to embed a video to web page. Video tag has same limitations too. Video tag supports only three video formats. In the first place it is MPEG 4 file with H264 video codec and AAC audio codec. This is a popular format that can take advantage of hardware acceleration supported by graphics chips. This format is often default format of recording video at devices. H264 is patented and it is free for no-commercial use. Secondly it is WebM file with VP8 video codec and Vorbis audio codec. WebM is released as an open source, royalty-free but still patented. Finally it is Ogg file format with Theora video codec and Vorbis audio codec. Ogg format is an open standard that is not patented and is royalty-free. Full-screen mode for playing video is not currently part of this standard, but application engine WebKit has same prototype of Full-screen mode. Adaptive streaming is not allowed or at least it is not mentioned at specification. The specification does not include (and evidently never will) Digital Rights Management (DRM). Advantages of video tag are more oriented on web browsers as on native applications like is Smart TV app. But HTML 5 has no proprietary functionality therefore standard JavaScript and CSS will work with it. [1]

HTTP Adaptive Streaming

Samsung Application Services contain features like HTTP Adaptive Streaming (HAS). Adaptive streaming allows users to enjoy seamless video streaming services over IP networks. HTTTP Adaptive streaming is an alternative to RTSP/RTP streaming and it is getting popular. Samsung Smart TV supports several protocols for HTTP Adaptive Streaming. In the first place it is MPEG-DASH (ISO/IEC 23009-1) protocol by Moving Picture Expert Group. Secondly it is HLS (HTTP Live Streaming) that is a HTTP-based media streaming communication protocol implemented by Apple Inc. as part of their QuickTime and iOS software systems. Thirdly it is OIPF HAS (Open IPTV Forum HTTP Adaptive Streaming). The last supported protocol is SmoothStreaming that is extension of IIS Media Services. SmoothStreaming enables adaptive streaming of media to Silverlight and other clients over HTTP. [2] [3]

3GPP decided to define a HTTP streaming standard in January 2009. Now it is the standard known as 3GP-DASH (Dynamic and Adaptive Streaming over HTTP). This standard convers the main aspect of HTTP streaming including storage, transport, and media rate adaptation for available link bitrates. First introduction of 3GP-DASH was at 3GPP 9 Release of PSS standard. Standard was finalized in March 2010 as Adaptive HTTP streaming. After finally 3GPP 9 release specification of adaptive HTTP streaming, the Open IPTV Forum (OIPF) adopted this specification and use it for their own Release 2 specification that was published in September 2010. [4]

Dynamic and Adaptive streaming over HTTP (DASH) consists of main two parts. First part defines the Media Presentation Description (MPD). MPD is used by clients to find the link to access media content. The second part defines the format of content in terms media segments as extensions to the 3GP and MP4 file formats. The default protocols of media segments retrieve HTTP. It is possible to use different protocols for media segments, but it must use URLs as unique identifiers. [4]

Media Presentation Description (MPD) is representation of XML file that is fetched at start of streaming used by HTTP. MDP may be updated in short time interval for more flexibility. MDP carries location and timing information to the client and playback the media segments like particular content. MPD consists of three parts. The main part of MPD is a Period. The period is larger piece of media, that is played segmentally. The each period may contain multiple different encodings of the content. Each alternative of content is called Representation. Different representation of content may be various at bitrates, resolution

or frame rates etc. Representation is a second part of MDP. Finally each Representation describes a series of Segments by HTTP URLs. [4]

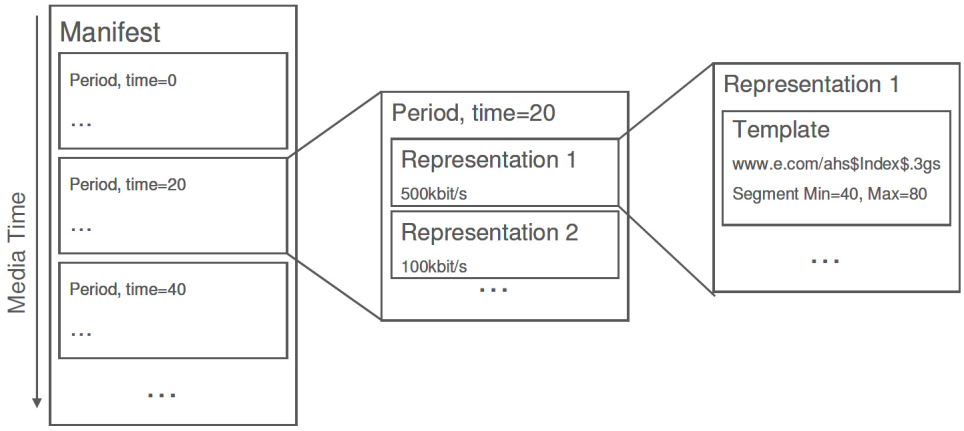

Figure 2: *Media Presentation Description layout*

The Media segments for 3GP and MP4 file formats are different to MPEG-TS media segments. The 3GP media file format is based on ISO format. A segment of 3GP file format for HTTP streaming is either as initialization segment or a media segment. The initialization segment contains configuration data. The media segment is a concatenation of one Segment type box and one or more movie fragments of media pointers. All of these segments together create a valid 3GP file. In case of MPEG-TS, the MDP syntax was reused with something profiling. There are added extensions for support of MPEG Transport Stream in media segments. Media segments are aligned with 3GP media segments. The MPD used MIME types that indicate MPEG-TS format. Two types of mimetypes are allowed: "video/mpeg" and "video/mp2t". [4]

Statistic of using

Usage of smart TV's features is not common at global view (not only Samsung Smart TV). While reading various articles and web forums we found out that using application on Smart is at low level. British survey shows that 25% owners of Smart TV never used their devices to connect to the Internet. Only one third (37%) of people planning to buy a smart TV said that connecting to the Internet was a reason for considering a purchase. However, 47% of all smart TV owners are using their device to access the Internet at least once a week. [5]

NPD Connected Intelligence Group's John Buffone wrote about the trends found:

"The Internet-connected HDTV screen has so far failed to break beyond the bounds of its TV-centric heritage, with little use for the big screen beyond the obligatory video services. The decision is not for want of application choice, but rather seems to be focused on how consumers are used to interacting with their TV. HDTVs, gaming consoles, Blu-ray Disc players, and other connected devices offer an array of applications... [but], in general, these have failed to resonate with the audience, not least because there are better platforms, such as the PC, tablet, or smartphone, for such services. [6]"

2.2 CDN

2.2.1 Overview

"Content Delivery Network refers to a large network of servers deployed across multiple networks in several data centers, often geographically distributed. CDNs are employed by companies across many industries to deliver HTTP content, rich media like streaming audio and video, and downloadable files. CDNs are optimized to provide higher speed and greater scalability and availability than is practical for a standalone server deployment." [7]

Nowadays the most popular terms are speed, accuracy, and availability of network-delivered content. These terms become absolutely critical at network delivery content. All these terms are measurable as Web performance. One of more things that helps to enhance web performance is Proxy server. Proxy servers cache web content, partially address the need for rapid content delivery. In this context, on users request for object, proxy servers check their cache memory when it contains requested object. If object is in the cache memory of proxy server and cache version has not expired then clients get a cached copy that typically reduces delivery time.

Web caching has some benefits, but unfortunately it has same drawbacks. First benefit is that it reduces bandwidth consumption, network congestion, and network traffic. The reason is that frequently requested web content is stored closer to client. Second benefit is reduction of time required to transfer web content from origin server to proxy server known as external latency. It is because proxy server has stored web content in its cache memory and it doesn't have to make request to origin server for requested web content. Third benefit, cashing improve reliability. It is because clients can obtain requested web content at time when origin server is unavailable. First drawback is when proxy server is not property updated, a client

can receive out of date web content. Second drawback is when the number of clients grows up then origin servers will become bottlenecks. [8]

Researchers have widely considered content delivery networks to be an effective solutions to reducing these disadvantages. CDNs are trusted overlay networks that offer high performance of delivery of common web objects, static data, and rich multimedia content. Requested content is distributing by servers, which are closer to client. [8]

Proxy servers and CDN essentially address two different issues. Internet Service Provider (ISP) use Proxy server to store the most frequently requested content at its local cache memory. Web servers use CDNs to store content, which is specified by administrator. CDN can store content, which Proxy servers can't. It can be secured content, streaming content or dynamic content. CDN consists of many distributed components collaborating to deliver content across different network nodes. There are four basic components which are collaborating to deliver. Fist, nonorigin, or surrogate, servers cache the origin servers' content. Second, routers deliver the client's content request to a suitable surrogate server. Third, various network elements distribute the requested content from the origin to the surrogate servers. Finally, an accounting mechanism provides logs and accounting information to the origin servers. [8]

2.2.2 CDN Concept

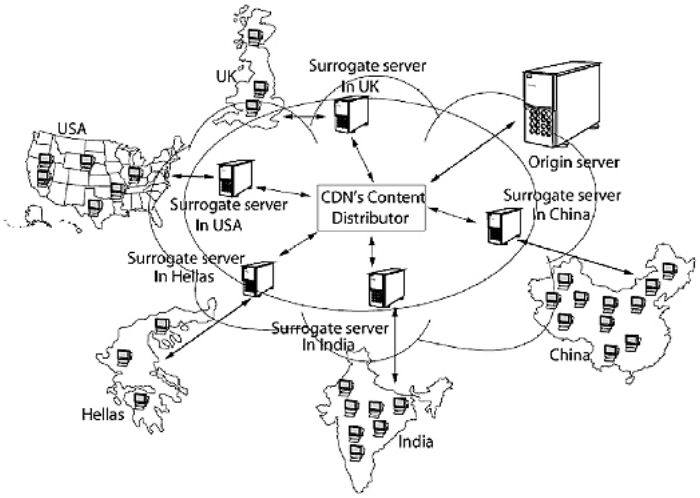

Figure 3: *Model of a CDN [9]*

Origin server

Origin server is a classic web server, that stores original web content. Origin server is maintained and updated by the enterprise.

Surrogate (nonorigin) server

Surrogate servers are the key part of the CDN structure. Surrogate servers characterize CDN structure. The main aspects are number of surrogate servers at CDN structure, where are surrogate servers located and which service clients will surrogate server request. CDN was created to enhance performance and quality of services (QoS) therefore also surrogate servers must be strategically placed across the Web. Strategic place for surrogate servers is important therefore there exist some algorithms for placement the servers for example: tree-based, greedy, clustering etc. As well as the location of Surrogate server is important number of Surrogate servers. CDN administrators must determine the optional number of them, because surrogate servers directly affects traffic cost. Optional number of Surrogate servers can be determined by two typically approaches:

- The single ISP use many of surrogate servers – at least 40 around its network. An ISP with global network can thus on adequately geographical cover without different ISP with their network. The most common policy is deployed one or two Surrogate servers to major country. In this case, the ISP use surrogate servers with large caches. A disadvantages of this approach are two. First, with larger caches at surrogate servers is larger hit ratio. Second, there can exist the surrogate server with large distance from the CDN's clients.
- The multi ISP use numerous Surrogate servers as many global ISP points of present (POPs) as possible. Location of Surrogate servers is important factor that has to be as close as possible to clients. Advantage of this approach is that CDNs can deliver content over different client's ISP. Some large distributed systems have more than 8,000 servers. A disadvantages of this approach are two. First, each Surrogate server handle fewer request, giving rise poor performance of CDN. Second, this case is more difficult to maintain that with single ISP approach. [10]

Different traffic volumes of sites seem to indicate that the single ISP approach works better for sites with low-to-medium traffic volumes and the multi ISP approach works better for high-traffic sites. But this is only estimate of performance.

The next critical issue is determination of the best Surrogate server. The "best" Surrogate server may not be the nearest server to CDN's client. On the most appropriate Surrogate

server impact several factors. In general, to choose the most appropriate Surrogate server is using abstract metric, topological proximity. Server is the most appropriate when it has the closest topological proximity to the visitor's browser. Topological proximity includes several criteria, such as speed, physical distance, data transmission costs and reliability. Some CDN implementation use the load balance between Surrogate servers multiple metrics such as proximity, server load and aggregate of the two.

DNS Lookup and Redirection

Primary issue of CDN is direct request from client to appropriate Surrogate server. This issue may be service two ways. First is DNS Lookup and second is redirection.

If client wants to access a web content, which is stored on Web server connected to CDN, first step is to resolve the server name portion of the URL to the IP address of server that stores requested web content. Domain Name System (DNS) translate URL to the IP address. Request is sent from client to the local DNS server. If local DNS server doesn't contain IP address of requested URL in its cache then local DNS sends request to the authoritative DNS server. Authoritative DNS server responses to local DNS server with IP address of Surrogate server. [8]

DNS redirection is based on mapping between a surrogate server's symbolic name and its numerical IP address. Local DNS server forwards request to the CDN's request-routing infrastructure (RRI) (Authoritative DNS server is controlled by CDN). RRI must service this request and choose Surrogate server which will deliver Web content to client. Choice of Surrogate server consists of several criteria such as topological proximity, packet loss or router hops (surrogate server to requesting entities). [8]

On the end this action local DNS server sends response to the user with IP address.

```
[NYC]% host www.symantec.com
www.symantec.com      CNAME      a568.d.akamai.net
a568.d.akamai.net     A          207.40.194.46
a568.d.akamai.net     A          207.40.194.49

[Boston]% host www.symantec.com
www.symantec.com      CNAME      a568.d.akamai.net
a568.d.akamai.net     A          81.23.243.152
a568.d.akamai.net     A          81.23.243.145
```

Figure 4: *Example of CDN mapping [11]*

Routing and Distribution

At the figure below is shown the content delivery process from a client request submission to the delivery of the requested object. If CDN providers support peering, the client request servicing procedure is more complex.

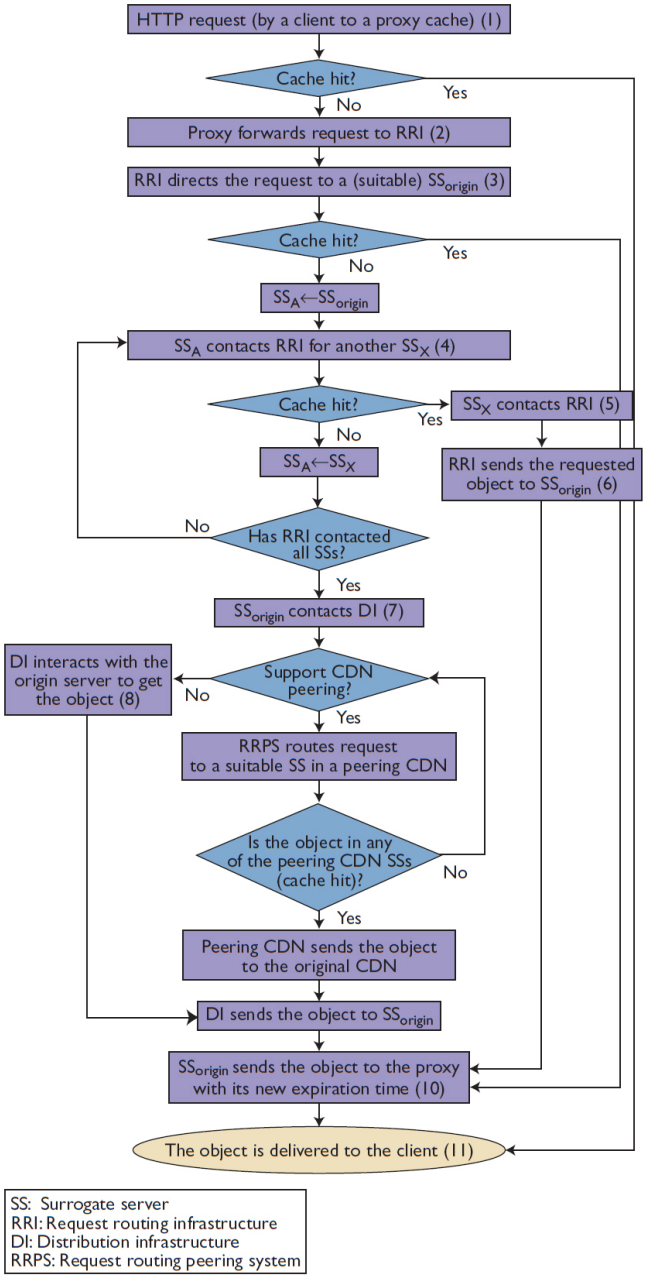

Figure 5: *Delivering the request object to the client. "Cache hit" indicates that the cache contains a current version of the object. [8]*

CDN peering

Peered CDN can deliver content over the other peered CDNs. The most common reason for peering CDN between different Internet Service Providers (ISP) is the improve performance and relatively low costs compared with solo CDN's. When client requests are received, it is

possible that RRI won't find requested object on any Surrogate servers. In this case, the RRI can send request for the object to the other peered CDNs that have connection to origin server. Among increasing number of Surrogate servers is increasing the possibility to be closer to clients. Peered CDN gives better throughput. Peered CDN has more possible ways to deliver content to clients (redundancy). [8]

2.3 Cisco Internet Streamer CDS

Cisco Internet Streamer Content Delivery Service (CDS) is proprietary content delivery network designed by Cisco Systems, Inc. It is a distributed network containing devices that collaborate each other to deliver multi-format content. All functions of Cisco CDS can be separated into four areas. These areas are Ingest, Distribution, Delivery and Management. On Figure 6 is shown abstract view of Cisco Internet Streamer CDS. There are major elements of a network. We can see Service Engines (SE), Service Router (SR), Content Delivery System Management (CDSM) and Origin servers. Service Engine in different literature can be named Edge server. Service Engine performs function of internet streamer and content acquirer that means it cares about ingest, distribution and delivery of content. The role of Service router is redirect client's requests. Role of CDSM is to manage the Cisco CDS. Shapes on figure indicates, how content flows from origin server to end users. Every delivering content is specified by delivery service. A delivery service is configuration defined by using the CDSM and it contains configuration parameters that define how content is ingest, distribute and what content will be delivered to client. Every delivery service belong to one content origin. Some of the primary parameters of delivery service are origin server, service routing domain name, service engines participating in the delivery service and Service Engine that is designed as the Content Acquirer. The Content Acquirer can be active only on one Service Engine in one delivery service. [33]

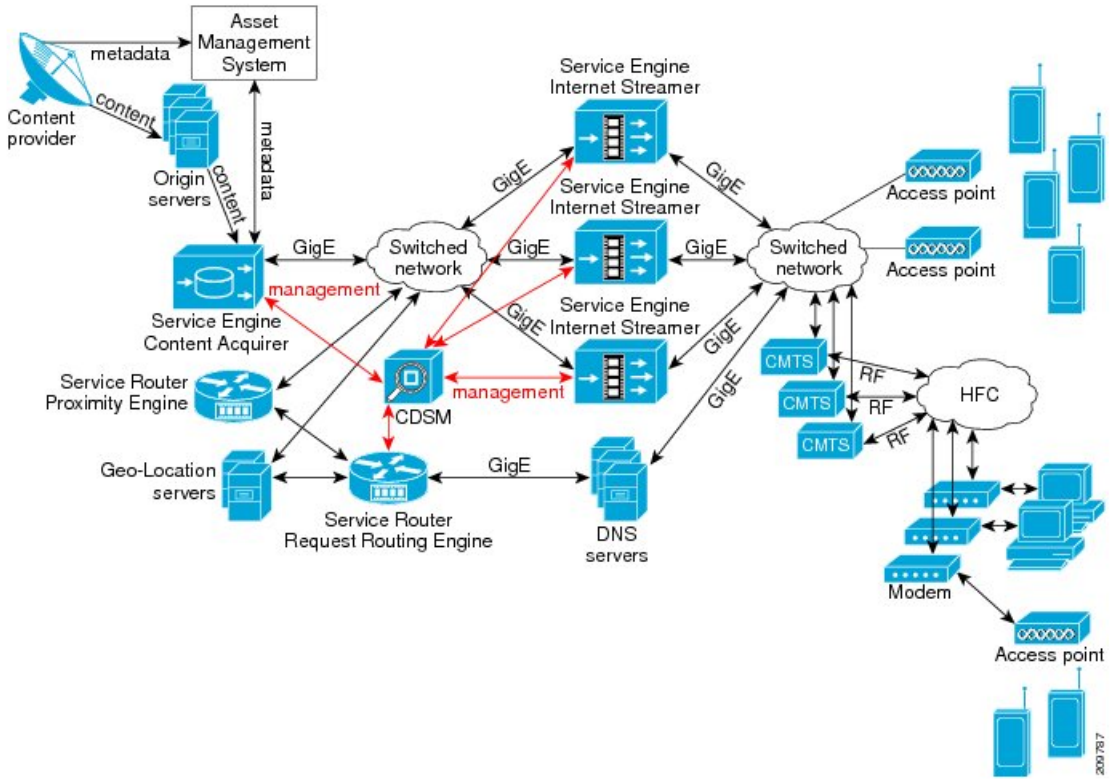

Figure 6: *Hight level view of Cisco Interneter Streamer CDS [33]*

Ingest device in Cisco CDS is Service Engine that is designed as Content Acquirer in delivery service. Cisco CDS supports following methods of content ingest: Prefetch ingest, Dynamic ingest, Hybrid ingest, Live stream ingest and split. Pefetch ingest is a method when Content Acquirer receives manifest file in XML format that contains information about files. Using this manifest file Content Acquirer ingests files. Files can be ingested using different protocols (FTP, HTTP, HTTPS, CIFS). Dynamic ingest is the method when content can be dynamically ingested. Dynamic ingest is triggered when Service engine detects missing requested file. When this content isn't frequently requested then will be deleted. Hybrid ingest method combine prefetch and dynamic ingest. This method use manifest file with information about files and pinned it to all Service Engines. Service Engines prefech this content when this content will be requested. In this case, manifest file can be used to specify explicit controls and policies for streaming the content. Live stream ingestion and spliting is used to increase the performance of life streaming service where is necessary to deliver content from one to many devices in the shortest time. [33]

Cisco CDS contains Content Delivery System Management (CDSM). It is a secure web browser-based user interface. This GUI allows an administrators to manage and monitor the CDS network that contains all devices registered to CDS. Administrator can create groups of devices where he can apply the same configurations. GUI allows to provide an automated workflow to update software in devices. [33]

Service Router has three parts: **Request Routing Engine, Proximity Engine, CDN Selector**. The Service Router can be configured as request router and proximity engine at the same time or Service Router can be configured as the Request Routing Engine. The Proximity engine is used for proximity-based routing. Proximity engine collects information about topology and path to locate the closest resource in the network. CDN selector provides third party streaming service selection. **CDN selector** is XML file that contains information about third party CDN and its location and content type that can be handled by this CDN. When CDN selector is enabled, the Service Router use information to determine which CDN from the CDN selector file will be used to handle client request. CDN selector file is an early field trial (EFT) feature. As we mentioned above, CDN selector contains location of CDN. If Cisco CDS has enabled Geo-location service to determine selection of third-party CDN, CDN selector use Geo-location server to identify location of client. CDN selector compares code of country in cdn selector file and code of country of client's request. If country codes match, then third-party CDN can be chosen. In other case it will be set as a default CDN. For selection of third-party CDN can be used type of content. It can be used together with Geo-location or it can be used separately. When CDN is selected then Service Router sends 302 redirection to client with translated URL. If selected CDN is Cisco CDS itself, then the request is handled over to the Request Routing Engine. The **Request Routing Engine** takes care about client redirection to the most appropriate Service Engine and controls load of the devices that allow automatic load balancing. Request Routing Engine is an authoritative DNS server that responds to any DNS queries. It supports three options for client to get routed to the Request Routing Engine and to the Service Engine. There are Router fully qualified domain name (RFQDN) redirection, DNS-based redirection and IP-based redirection. On Figure 7 is shown RFQDN Redirection of client request. Client sends a request for a "video.cdn.com". Browser recursively resolves domain name used by DNS proxy. Service Router is an authoritative DNS server for requested URL. It returns a record with IP address of Service Router. Client sends request directly to Service Router that returns 302 redirection to "se1.se.video.cds.com". Client sends request for new domain "se1.se.video.cds.com". As we mentioned above, authoritative dns server is Service Router

that resolves requested domain name and return A record that is IP address of Service Engine. Then client knows, where will be handled and he can send request to Service Engine. [33]

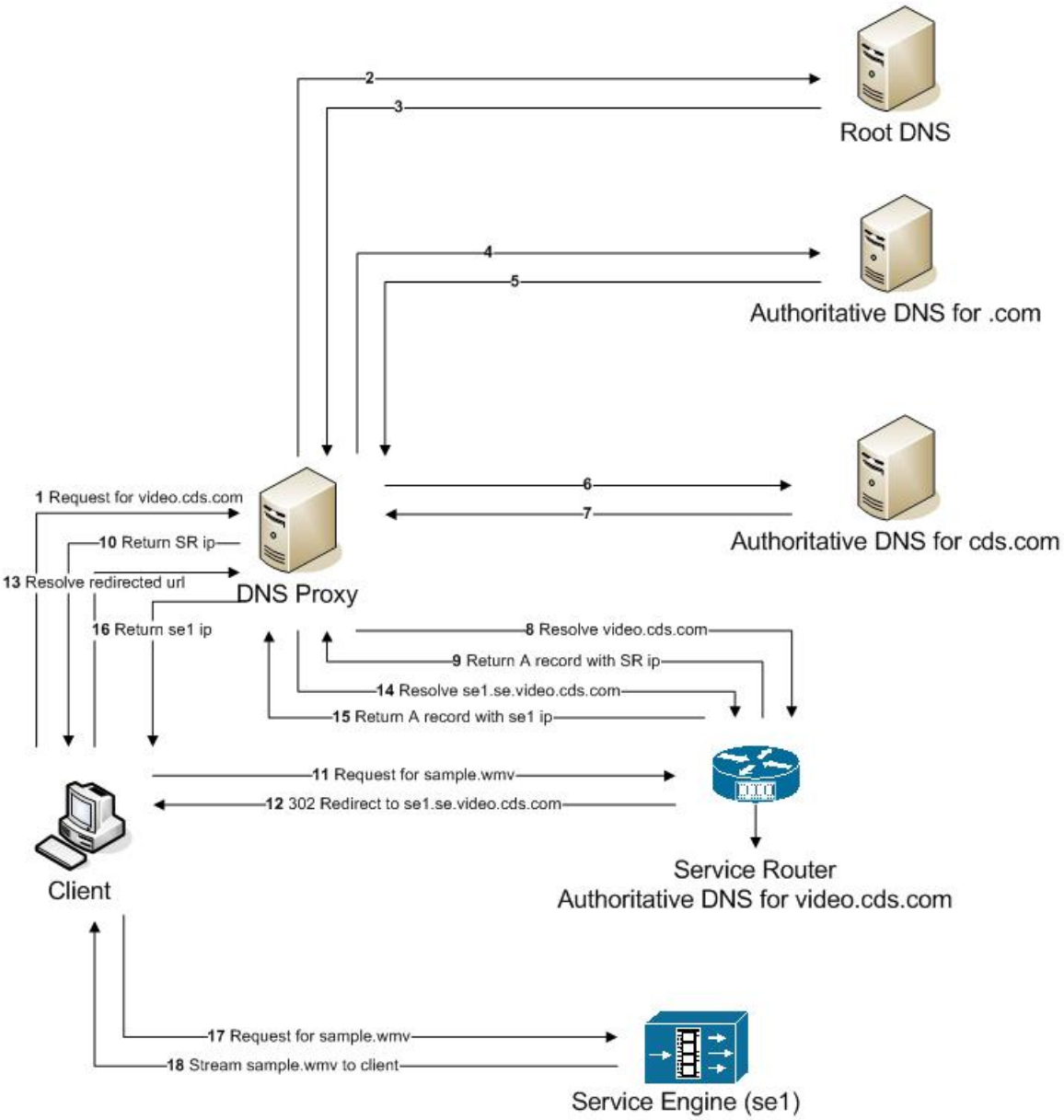

Figure 7: *Workflow for RFQDN Redirection [33]*

On the Figure 8 is shown DNS based redirection. In this case, client is directly routed to Service Engine without any 302 redirection. DNS proxy asks authoritative DNS server for "video.cds.com" to resolve domain name. This server is Cisco CDS Service Router and it doesn't return its own IP address, but directly A record of Service Engine which will be

chosen based on load, location and some other heuristics. After this response client can make request directly to Service Engine. [33]

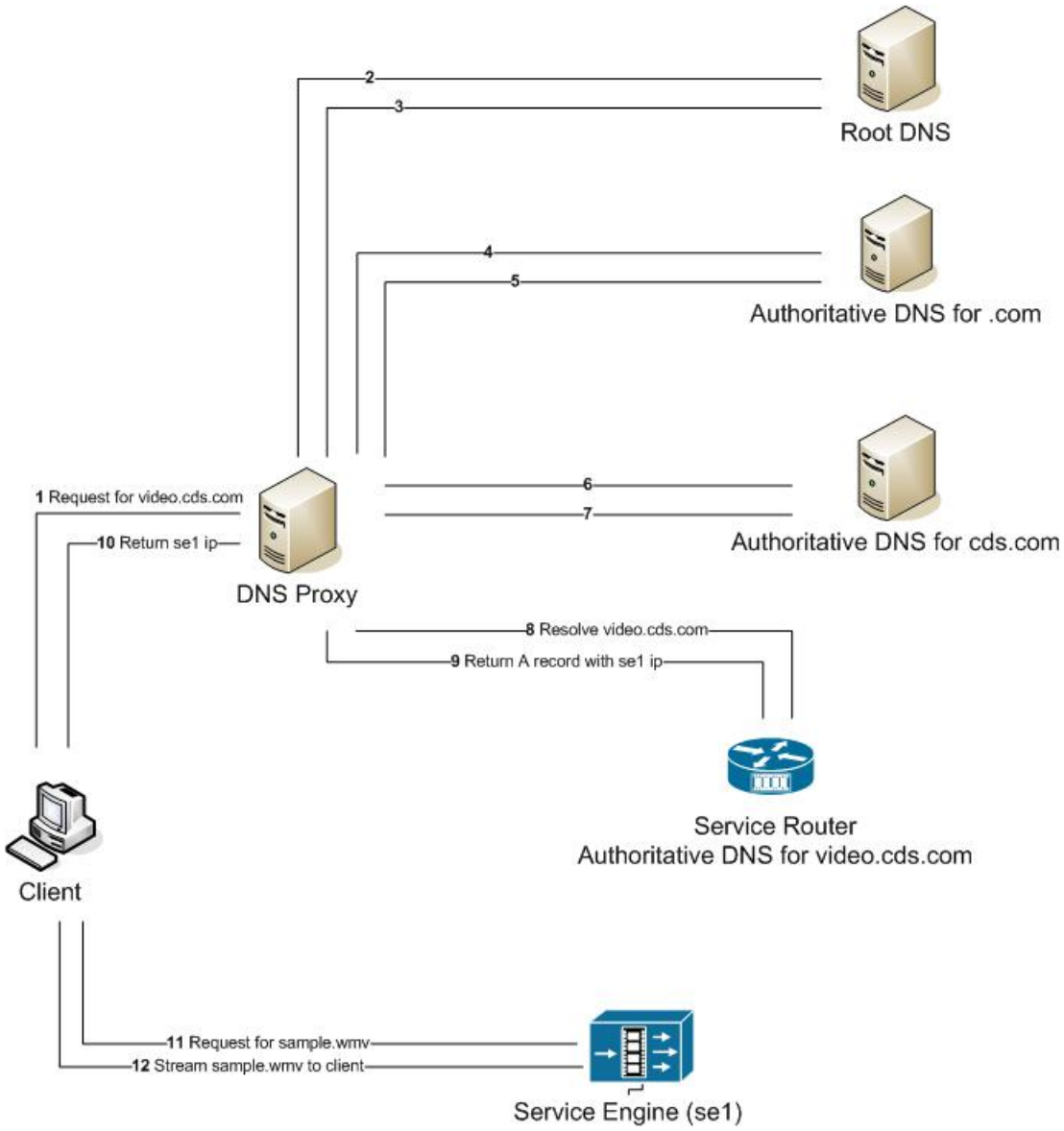

Figure 8: *Workflow for DNS based Redirection [33]*

Service Router is used to determine Service Engine **Coverage Zone** file. Coverage Zone is XML file that contains footprints that can be handled by Service Engine that have assign this file. Coverage Zone file can be assign globally to all devices or every device can assign own Coverage Zone file. On Figure 9 is shown example of usage Coverage Zone file and how it cover area of clients with two CZ files. [33]

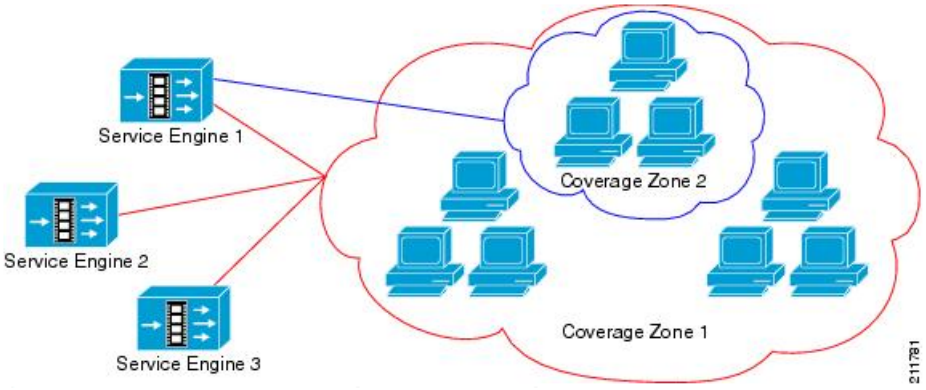

Figure 9: *Example of used Coverage Zone file [33]*

2.4 Android

2.4.1 Overview mobile platforms and statistics

The most popular devices of nowadays, that people carry an everyday are smartphones and tablets. These devices simplify daily work in variously sectors. We can use them for calling, sending text messages, sending E-mails, browsing Internet, using Internet banking or much more. Without all of these functions or gadgets, a lot of us cannot imagine everyday life. People have to become accustomed to using these devices. Device is only interpret of logic structure of software. Smartphones of nowadays are using three primary operating software. They are iOS by Apple Inc. Android by Google Inc. and Windows Phone (currently at version 8) by Microsoft. Based on statistics, which were created by IDC (International Data Corporation) and presented by Time Tech, Android is the most used operating system in smartphones over the world. Android have 70.1% world market. If we will focus on market with tablets, 7" tables with Android decrease lead of tablets with iOS, but iOS is still the most popular operating system in tablets over the world with 53.8% world market. [12]

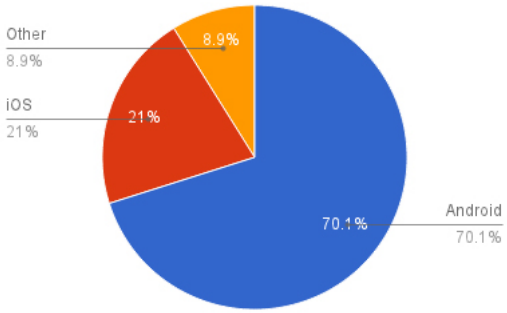

Figure 10: *IDC worldwide smartphone shipments, 4Q 2012*

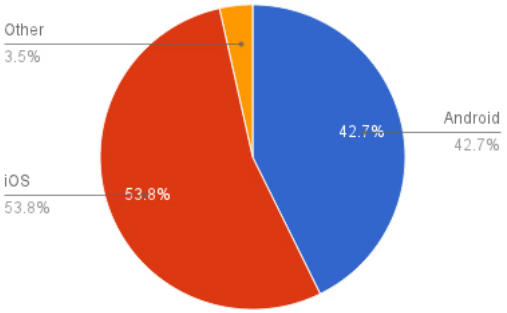

Figure 11: *IDC tablet shipments, 2012*

With increasing number of smartphones and tablets (mobile devices) is increasing the number of mobile applications too. There exist a studies, that are focused on spent time with activity on mobile device per day, how many time we spend for different types of applications or how many applications we used in mobile devices per day. For example: Flurry Analytics per past five years worked with tens of thousands of developers to integrate their analytics and add platform into mobile applications by the developers. Today, Flurry Analytics have 2.8 billon application sessions per day. [13]

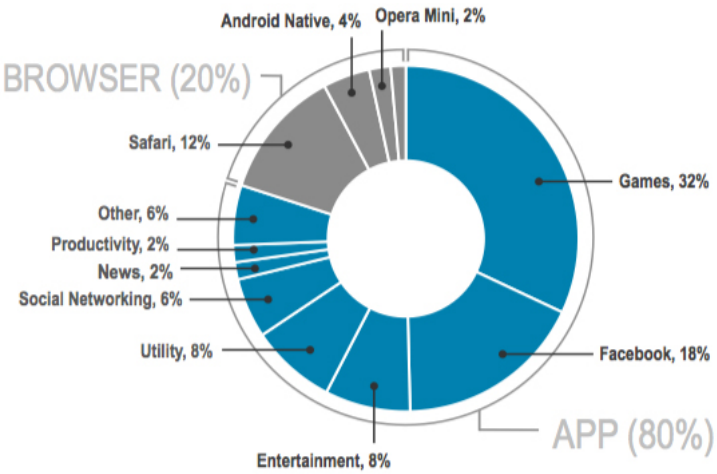

Figure 12: *Time spent on iOS & Android connected devices*

Analyzes by Flurry Analytics follows, so the U.S consumers spends an average of 2 hours and 38 minutes per day on mobile devices and they used average 7.9 applications per day. All these numbers are growing up every year. With these numbers arises bold statement such as: *"It's an App World. The Web Just Lives in It"*. [13]

For our master thesis are these facts more important and bring an idea of development for mobile device.

Android is distributed to mobile devices from version 1.6 with codename Donut. This version use mobile device' users less than 0.1% now. The newest version of Android is 4.2.x with codename Jelly Bean. Development from version 1.6 to 4.2.x lasted about four years. Android platform has five versions with different codename. The most using versions of Android devices are 2.3.3(-7), 4.0.3(4) and 4.1.x. All of them have the market coverage of around 30%. We will focus on the version 4.x.x because this version contains low level support streaming media. [14]

2.4.2 Android 4.0 Ice Cream Sandwich

Android under version 4 contains two code names - Ice Cream Sandwich and the newest Jelly Bean. Both code names conceal the same base.

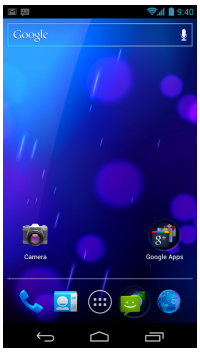

Figure 13: *Android 4.0 Ice Cream Sandwich*

Android 4.0 brings a unified UI framework that lets developers to create application for mobile devices independent on mobile platform such as smartphones, tablets or more. Unified UI framework in Android 4.0 means new UI tools, simplified code and resources, consistent design practices, and streamline development across Android powered devices. Unified UI framework is very helpful with development on more types of devices. If we will decide for development for Android platform, we have to create application for smartphones, tablets and set top boxes. [15]

The main part of Android features are Media and Connectivity for us. This part contain all things needed for mediation multimedia content at mobile devices.

Architecture

Android platform adheres to the concept "write once, run many". This ensures that application, which is developed and tested on one Android platform of given architecture, shall run identically on another Android platform of different architecture. This concept is not new. JAVA programing language was created on this concept. Generally, Java applications tend to run slowly like applications written in C++/C. Android leverages Java only partially. Android applications are written in Java, but they rely on application framework, associated libraries and runtime which are written in Java and mostly C++/C languages. The link between Java and C++/C language is premised on performance, with "slower" code written in Java and "faster" code written in C++/C. [16]

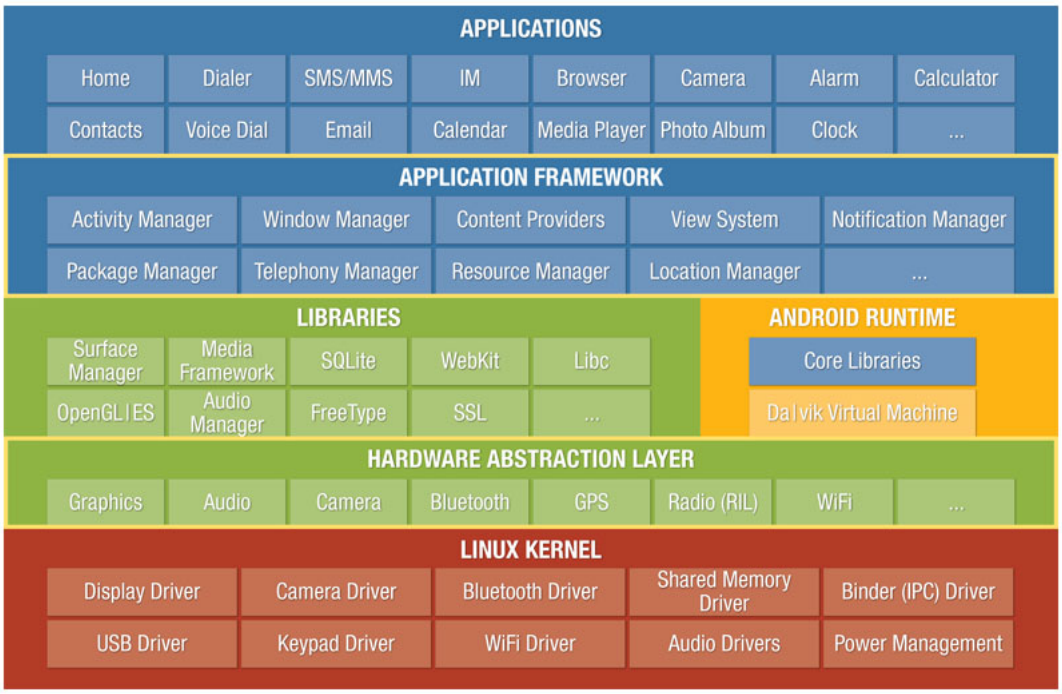

Figure 14: *Android architecture*

Low-Level streaming multimedia

From version 4.0 Android platform provides direct and efficient path for low-level streaming multimedia. These features provide options for developers to maintain complete control over media data before passing it to the platform presentation. For example, application retrieve media data from source, apply proprietary functionality, and then pass media data to the platform presentation. Application can send media content to the platform as a multiplexed stream of audio/video content in MPEG-2 transport stream format. The platform de-muxes, decode, and render the content. The audio track is rendered for the active audio device while the video track is rendered for either a Surface or a Surface Texture. Application can add effect on each frame using OpenGL, when video track is rendering. [16]

Low-Level streaming multimedia is support thought native API based on Khronos OpenMAX AL 1.0.1. The API is implemented on the same underlying services as the platform's existing OpenSL ES API, and developers can use both APIs together. [16]

OpenMAX AL includes the ability to create and control player in multimedia applications. OpenMAX AL provides standardized interface between an application and multimedia

middleware. It is applicable to all applications where multimedia performance is a critical issue. [17]

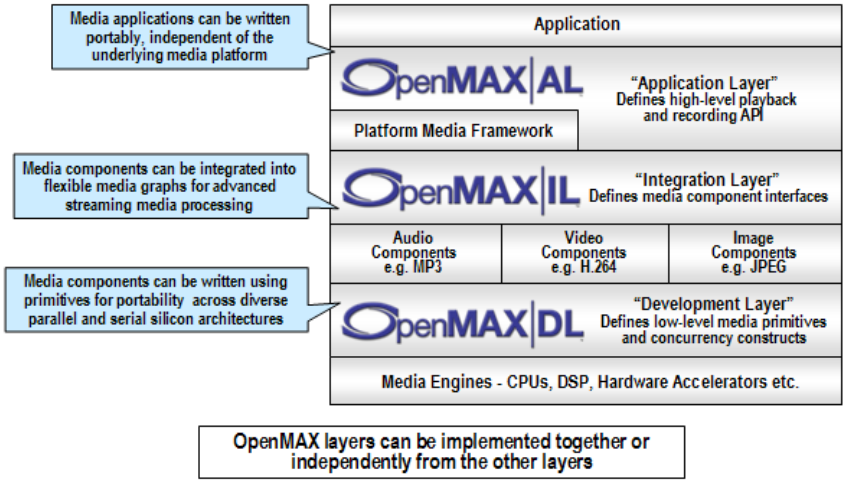

Figure 15: *OpenMAX Media Portability Library*

Android Supported Media Formats

Android 4.0 supported a few network protocols for transporting media content. Following protocols are supported:

- RTSP (RTP, SDT)
- HTTP/HTTPS progressive streaming
- HTTP/HTTPS live streaming (MPEG2 TS media files only, protocol version 3)

Media format, which is supported into the Android platform for video are:

- Codec H.263, (3GPP, MPEG-4)
- Codec H.264 AVC, (3GPP, MPEG-4, MPEG-TS)
- Codec MPEG-4 SP, (3GPP)
- Codec VP8, (WebM, Matroska)

[18]

2.5 Existing solutions - Google Play Movies and TV

Google Play Movies is a part of the Google Play Store. We can find movies between standard Play Store items. Movies are relative new in Play Store and this is reflected on quality of

service. Movies have only five categories. The interface is just simple. Interface is limited to the five categories, section showing top selling movies, featured movies and so on. Every movie can by placed at more than one category. Some of these movies are placed in wrong category. Searching movies based on their genre, release year, price, resolution or some other factors don't work correctly. It is disappointing to see the biggest search engine work incorrectly. [19]

After start application Google movies you can rent or buy movie. Client doesn't has any offer to use filter on movie only has five categories which contains messy content. Some of the newest movies can clients only buy and not rent, but on the other hand some movies can clients only rent and not purchase. Google play disposes an offline mode feature. Client can after purchase or rental of movie download it to device and watch it without Internet connection. Clients have two options of rental movie. Client can choice rental for 48hours or 30days during when can watch movie again and again. Client can rent movie in two different qualities - HD or SD. Under HD quality you can find resolution 1280 x 720 pixels (720p) but you can find 1920 x 1080 pixels (1080p) too. User profile, which is used at this application is standard Google account. Yo are can't login on two devices and watch the same movie on both device. It is impossible. [19]

Google play is relatively new service. When we wanted to test this application on our device we were stopped by the page with attention, which informed about missing support of service in our country. We have to rely on opinion from the resources. [19]

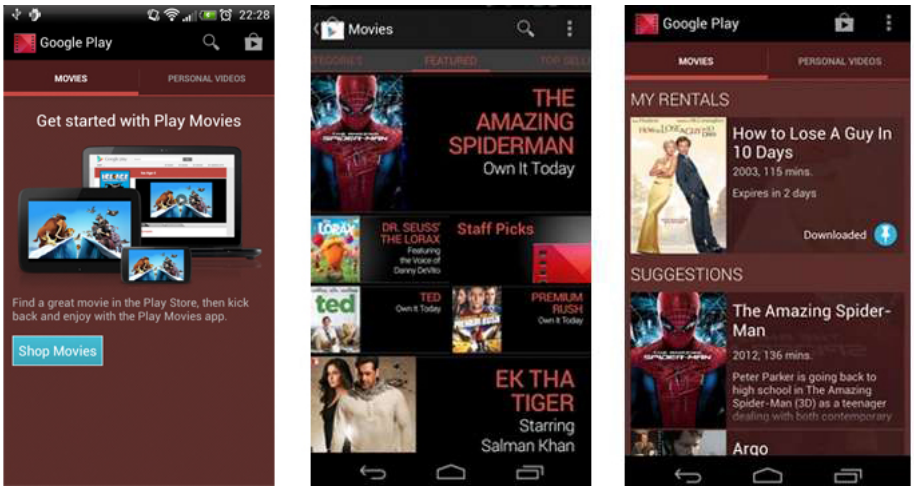

Figure 16: *The screenshots from Google play movie service*

2.6 Existing solutions – LOVEFiLM

LOVEFiLM belongs to Amazon Company. This service is accessible from web page, mobile application or application for Smart TV. Mobile application was developed for iOS and Android platforms. We tried mobile application for Android platform. Applications provide users a few features. Users can search and browse entire catalog of Films, Games or TV. Catalog of films contains a lot of genders and hot lists. Hot lists contain new releases, most popular and coming soon lists. Catalog of Games and TVs has got a lot of genders too. Users can use feature My list (or Rental list). User can manage own Rental list (add, delete). Application allows to vote for every title. User can see rating of every title in the part overview of title or at list of titles. After click on some titles at list overview of title is shown. In overview we can find detailed information about title (cover picture, rating from users, director, short description, cast, genre, release date, production year, runtime, format and studio). The most interesting from all information in overview is trailer and reviews. Under reviews are user's comments for the title. Every user can watch all of these information for free. There are two ways for watching films. User can rent or buy a title. In both cases there exist options to watch it online or from DVD media. If user rent a disk, it will be send first class mail to user and user can send DVD disk back after watching. If user wants to watch title now, then can use options with online watching. But LOVEFiLM did not support offline watching as Google Play Movie & TV. [20]

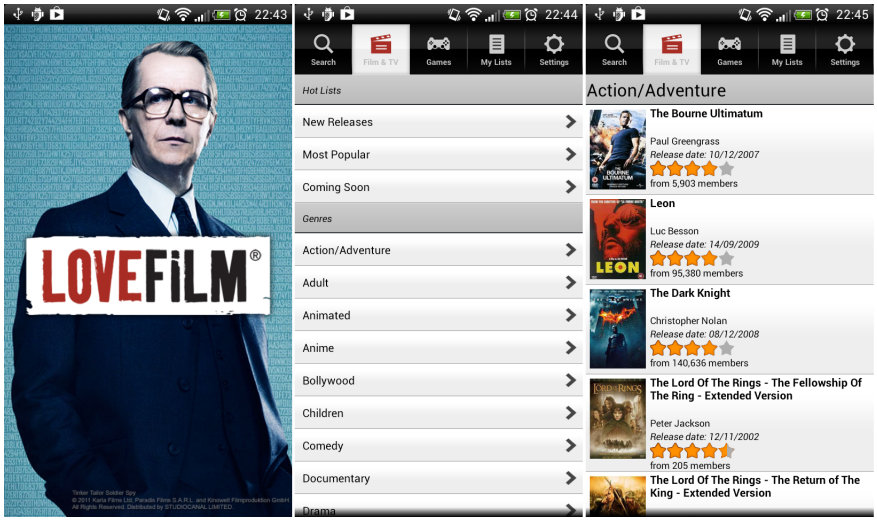

Figure 17: *The screenshots of LOVEFiLM service*

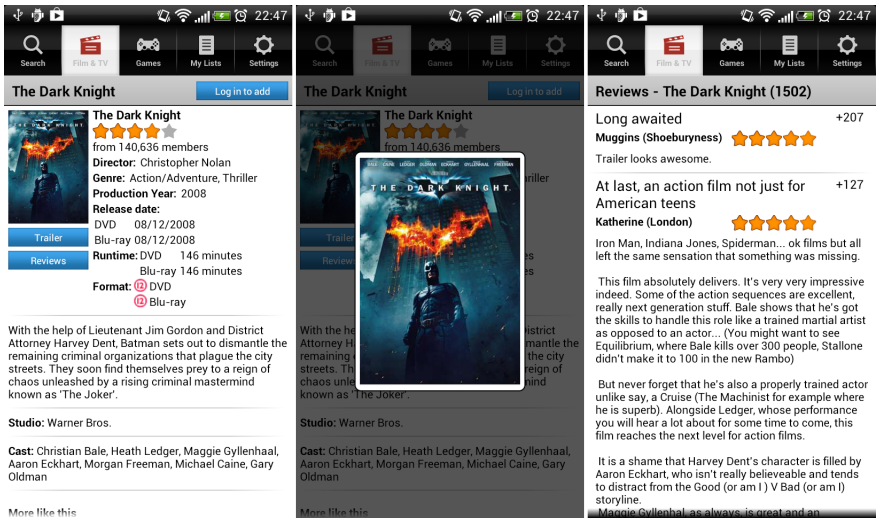

Figure 18: *The next screenshots of LOVEFiLM service*

2.7 Existing solutions – Cisco VDS Service Broker

Cisco Videoscape Distribution Suite (VDS) Service Broker (SB) is CDN selection product. Cisco VDS SB provides selection appropriate CDN based on Geo-location, content type, business rules, time-of-day, CDN status and resource availability, client type and other. Cisco VDS SB contains Service Broker and VDS manager. Service Broker is request routing engine that routes request from user based defined criteria to chosen CDN. This method allows interconnection between Cisco CDS and third-party content delivery networks. Videoscape Distribution Suite Manager is running on one or more SB devices. VDSM is web based application with graphical user interface to setting up SB. VDSM contains web service API for provisioning and monitoring Service Broker. CDN selection interfaces operate in three modes. These mode are HTTP GET, DNS and Web Service. HTTP GET mode used standard HTTP GET protocol to invoke CDN selection and response is redirect. HTTP GET request contains BFQDN and path to the manifest file or content fragment. The HTTP header can contain additional information. The VDS-SB can rewrite URL if selected CDN need different structure of URL. DNS mode operate as authoritative DNS server. The VDS-SB is invoke if it is authoritative server of requested domain name. CDN selection in DNS mode is limited. DNS mode can use only location, percentage allocation, broker FQDN and CDN status. CDN is selected based only information contains in DNS request. Response on request can be HTTP 302 redirect or DNS to the appropriate CDN node. Web Service mode performs

communication with server side application. Request can contain IP address of client and manifest file or content fragment. Response message is XML file. It contains URL to the content on selected CDN. As we mentioned above, VDS-SB can rewrite URL since some CDN support is different to structure of URL. [30] [31]

2.8 Conclusion

Assignment of Master thesis was enough change. Based on these changes we have to renew our analysis in last stage of project. After second stage we change assignment and we focus deeper to content delivery networks. Concretely we focus on interconnection of CDN. After first stage of project, our analysis contains view on functionality and future disadvantages of Samsung Smart TV platform. Samsung Smart TV is interesting idea but using it is too hard for users. We think user experience is on very low level and we think that future of home media is different way. Therefore our next focus was on Android platform. Market contains devices like as Android Smart TV Box. This devices aren't often used in home but Android platform is used enough in devices and it is well known for people. Therefore we think that way of home media will walk over Android Smart TV boxes or exist concurrency like as Apple TV. Our analysis contains overview for content delivery networks too. There we met with devices in content delivery network, their roles and structures. In the end of analyses we found existing solutions of applications. But after second stage of master thesis we renew analysis and we added analyze of Cisco Internet Streamer CDS that we finally use for testing our implementation. We added existing solutions of interconnection that CDN provides by Cisco Systems, Inc.

3 System Design

3.1 System design for delivery of multimedia content

Goal of the master thesis is to create system for video content delivery that will have guaranteed quality of delivery. For this purpose we will design and create library of movies. This will cover all main points of master thesis. Library will consist of three main parts. First part is web server. Web server will store all media content and scripts for service requests. Second part is an application, that makes media content accessible for end users. The last part is network, which will be made by several servers. These servers will be configured as a content delivery network. On Figure 19. is representation of three main parts of our designed system.

Figure 19: *Three main parts of designed system*

On the base of our analyses of Samsung Smart TV and Android, we will have to make important decision. We will develop application movie library under Android platform. There are several reasons. The main reason is usage of Smart TV, which is very poor. Next reason is feasibility of devices based on the Android platform. We can find android platform on smartphones, tablets or for us very important set top boxes. With set top box we can create full-fledged application for television but thanks to Android platform we can use our application in smartphones and tablets too. These features rise up support of our application in multiple types of devices. Android platform provides an implementation advantages because code can be written on lower level like code written for Samsung Smart TV.

3.1.1 Specification of requirements

There are features which our system has to have.

- Storing multimedia content at server
- Storing information about multimedia content in database
- Storing data about user in database

- Provide movie library for users
- Provide movie detailed information for users
- Provide search of movies for users
- Provide watching of movies for users
- The history of rented videos
- Sorting of movies to categories
- Rating of movies
- Billing of movies for users
- Transfer data between server and client's application using be CDN
- Transfer multimedia data using one of HTTP streaming protocols

3.1.2 Application

The main functionality of our application will be mediation of multimedia content to the client. In the analysis we examined the existing solutions like Google Play Movies & TV and LOVEFiLM. Thanks to these applications, we are able to inspire our design and our initial thoughts shifted to higher level. The user will have the multimedia content sorted into categories. He will be able to search each category and filter the multimedia contents. Except the basic categories based on genre, user will also have a special category, where the multimedia content will be sorted by Most Rated titles and according to the latest titles added to the system. Users will be able to search by the title name and will be able to combine it with other parameters that will contain the title (e.g. genre rating, length). User can see history of his purchases. The multimedia content can be available for free or will be subject of billing. Each multimedia content that will be available free of charge will be rented and it is possible to be also purchased. Rent of title will be for a fixed period that may be set in our system.

The application will be developed for the Android platform. The programming language will be Java. Applications will be fully adapt and intuitive for easy use as the smartphones or tablet as well as on TV via set top boxes. In our system, we focus primary on quality of delivery of multimedia content and therefore we decided to use HTTP streaming. Final form of applications will be possible to install on your Android device with the version 4.x.x or plus with .apk file.

3.1.3 Origin server

Origin server will be our main server, that will always contain current data and the main server to handle requests from the client. Origin server will also contain the main database. Database will store user's information and their watch history, all data for multimedia content. The scripts will create responses for request by users from database.

Communication with the server will be established through the HTTP protocol. All communications including multimedia content will flow through the HTTP protocol. Scripts that will handle requests will be written in PHP. On the server will run Apache2 and PHP will be capable of processing PHP CGI. Preclude the use of supporting tools for faster processing of PHP scripts such as HIP - HOP for PHP. PostgreSQL database will be used for storing information on the server.

3.1.4 CDN

Our core system will provide the high power increase of data transmission with the quality of the content delivery network. During our analysis, we were looking for the possibility of creating a CDN network. There are several tools that set the server to operate as the origin server in the CDN. For such purposes we used an Apache module with which we will be able to set the server to operate as the original server. No such instruments would we still get to the level CDN network simulation and simulation of a multimedia content. We will work with a real CDN network that has servers are in Bratislava.

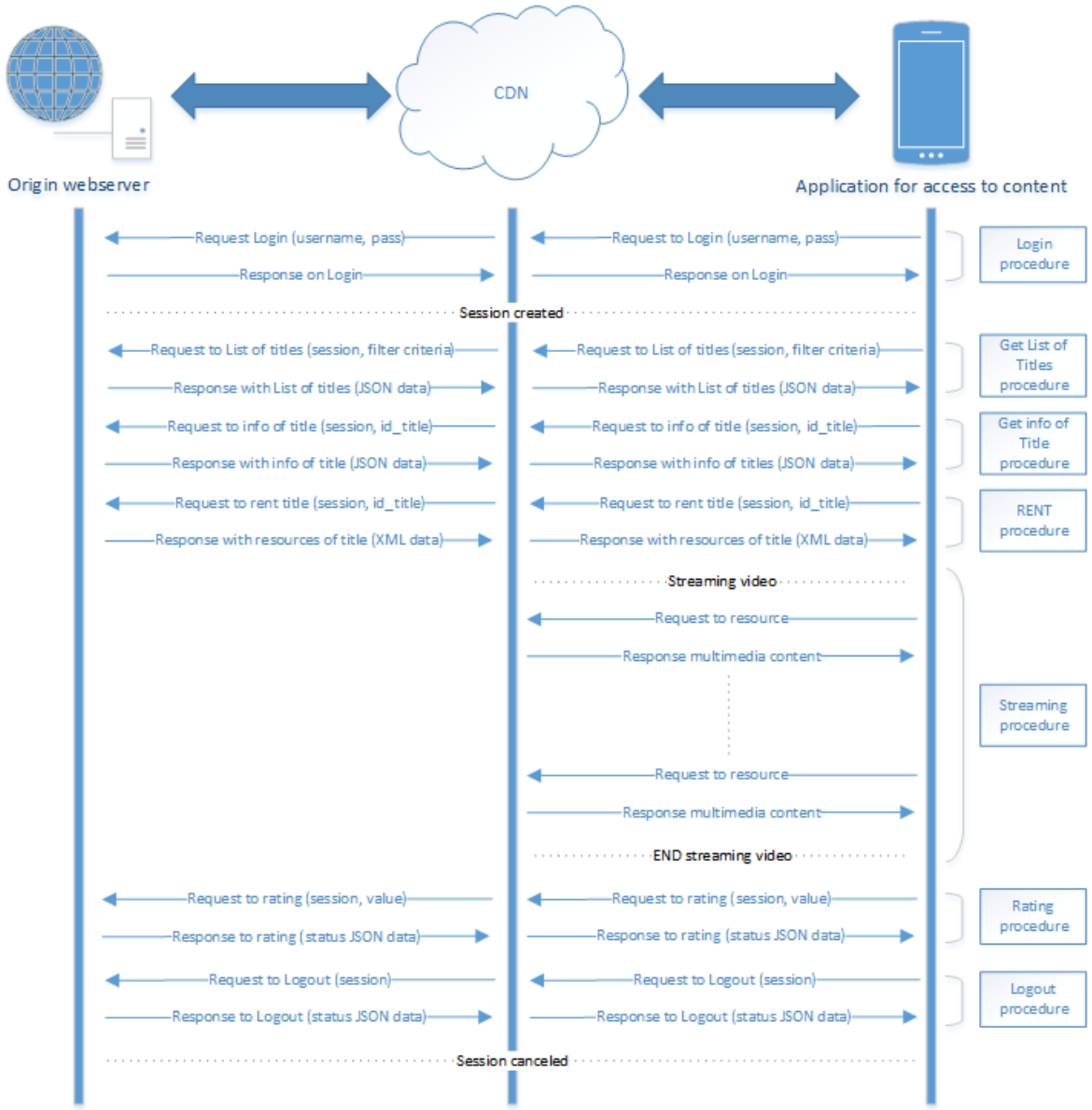

Figure 20: *Base model of interaction application with origin server over CDN*

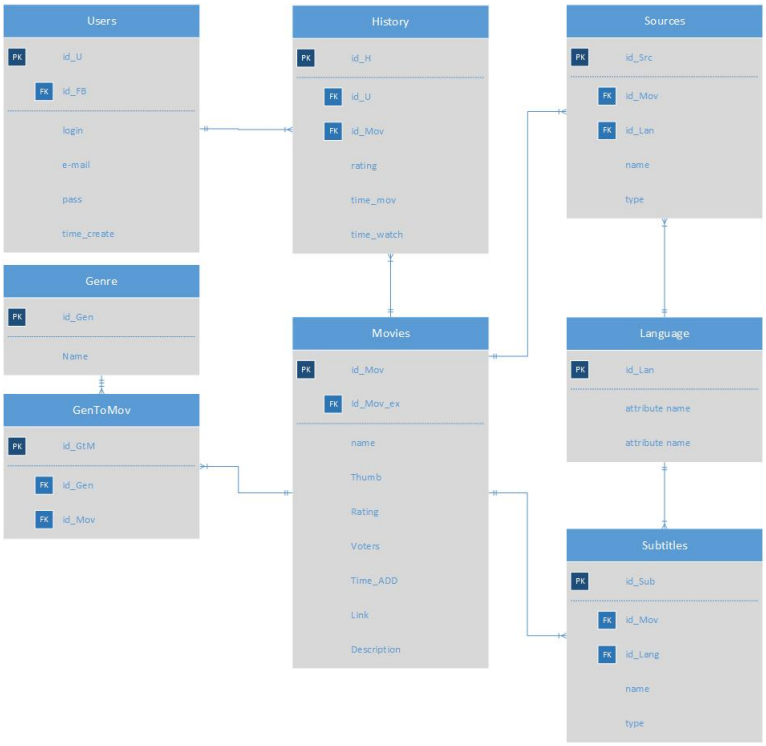

Figure 21: *Prototype structure of database*

In Diploma project 2 we have available Content Delivery Network by CISCO (Cisco Internet Streamer CDS). This CDN has great performance. Our original idea was to create set of tests and use it for testing the CDN. But high performance of the CDN would cause that our tests would not be beneficial. Then we are familiar with problematic of interconnection of CDNs. After that we focused on increase of quality of content delivery by cooperative two foreign CDNs.

3.2 Introduction to CDN interconnection

Purpose of this master thesis is delivery content to end users with guarantee of quality of service for using content delivery network. In this section we focus on describing and designing our solution. We explicate idea about interconnection of two different branding Content Delivery Networks and their cooperation together with some content. As we describe higher, Content Delivery Network cooperates with the Origin server and handles its requests to content.

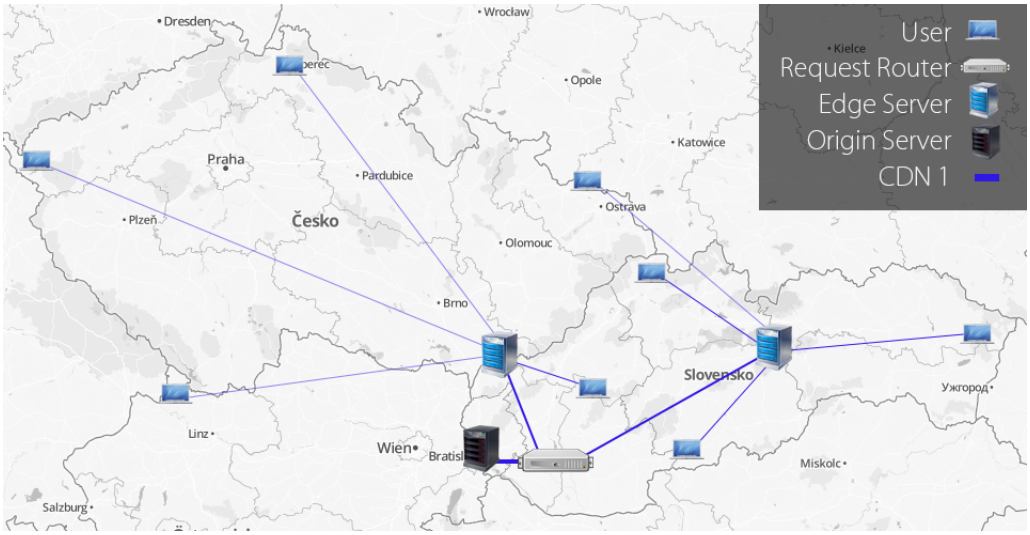

Figure 22: *Show topology of streaming content from origin server to users*

Consider a situation in which some providers provide live stream video from some special event like is ice hockey match in some country and provider wants to delivery this stream to other country. Providers have Origin server that cooperates with one CDN and CDN has Edge servers only in the same country as live stream origins. This case represents figure 22. As we can see in this case geological location of origin server is in Bratislava. Request router of CDN that handles requests for origin server has geological location in Bratislava too. One

Edge server of CDN is near Bratislava and second Edge server is the middle of Slovakia, far from Bratislava. These two Edge servers store content, which will be delivery to end users. If users will watch live stream only from Slovakia, their distance from edge server will be short and delay to server will be probably low. But in the situation where a lot of users which want to watch the live stream of hockey match coming from e.g. Czech republic this may be uncomfortable. With increasing count of active users increases delay for requests too and it can lead to the lost of connection. Among increasing distance between users and edge server increases delay too. This situation can have a few solutions. First, provider of Content Delivery Network can add a new Edge server in this case to Czech republic. Second, Provider of origin server can make a new deal with different provider of content delivery network in the Czech republic area.

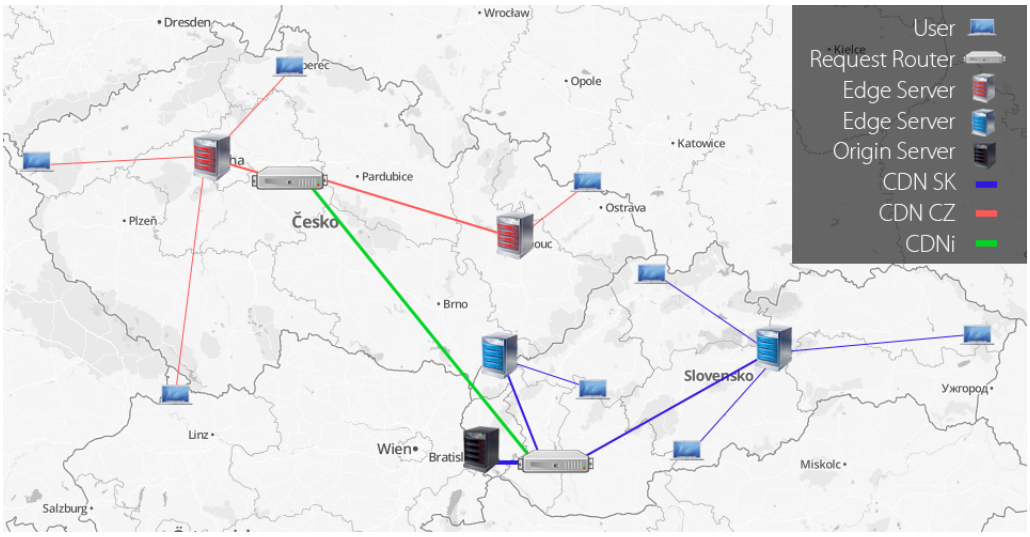

Figure 23: *Show topology of streaming content from origin server to users*

Both of these solutions are uncomfortable.In this case come idea of union of two content delivery networks to cooperate together as one big CDN. This idea has some hitch. The main hitch is that all solutions of cooperate two content delivery networks as one are only in theory. For this connection doesn't exist any tool. Figure 23 demonstrates situation of topology if we use two content delivery networks to delivering content to end users. The point of this topology is to delegate distribution of content to different area by different CDN that are appropriate to area. Request router from Czech republic communicates only with Request router at Slovak area, no with origin server. Request router in slovak area collects content from origin server and distributes it to the edge servers and other CDNs. The problem is

in green line on Figure 23 that is used to connect two request routers between two CDN. This is the main issue of our master thesis. Design and creation of connection between two CDNs. In April 2013 was released ETSI document with attributions TS 182 032 that focus on interconnection of content delivery networks. This document describes how two CDNs can establish connection and share content between them, but all of these ideas are only on theory level. This document naming this connection as CDN interconnection alias CNDi. We would like to use this idea of CDNi and design and develop prototype of CDNi. [21].

3.3 Content Delivery Network interconnection CDNi

3.3.1 Overview

Document ETSI TS 102 990 [22] presents technical requirements for CDNi services. All of these requirements were created based on the use case where are described at ETSI TS 102 990 in annex A [22]. In this clause we focus on CDN relationships, functions and concept of domains. These parametres are represent in Figure 24. As we can see on Figure 24, communication shall initial user with his demand on some kind of content (light grey shape from customer to content provider). Whereas content provider has agreement with CDN provider, which will be delivering his content that content provider delegate user request to serve on CDN provider. Request on content is now in CDN domain. CDN domain contains two content delivery networks connected with CDN interconnection (CDNi will be detailed describe later). One of them, closer to origin server (only in this case on figure, it is not obligatory) is Ingesting CDN (upstream CDN) and second is Delivering CDN (downstream CDN). Ingesting CDN is entity that ingest original content from origin server and bring it to CDN domain. Through CDN interconnection is possible to controll distribution of content from ingesting CDN to delivering CDN. Delivering CDN is entity that delivers content to consumer domain.

These are summary of main entities:

- **Consumer**: entity, where content is consume
- **Delivering CDN** entity, which delivering content to consumer
- **CDN interconnection** entity, which connect Delivering and Ingesting CDN
- **Ingesting CDN** entity, which ingest content to CDN domain
- **Content provider** entityt, which own origin content

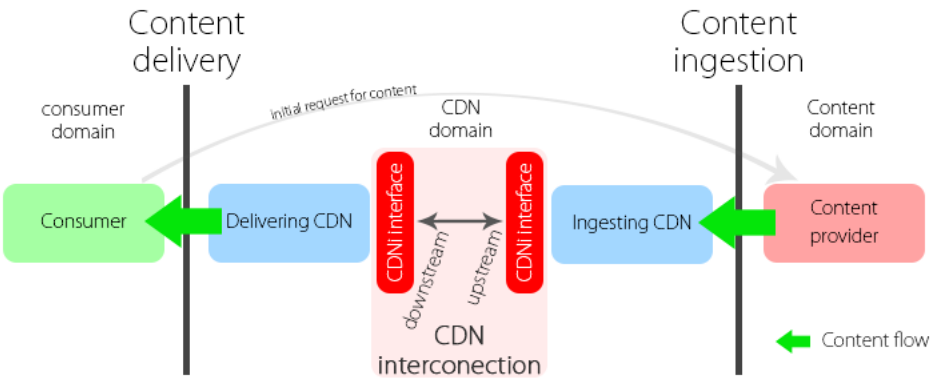

Figure 24: *Show relationships between CDN provideres, their functions and concept of domains*

In Figure 24 we shows high level principle of delivering content via CDN interconnection. In previous scope we wrote about use case with ice hockey match. Figure 25 shows principle of delivering content via CDN interconnection on use case with ice hockey match for better understand principle CDNi. In the Figure we replaced entities with real devices which topology was shows in Figure 23. Now we can see which devices of both CDNs cooperate on delivering to consumer.

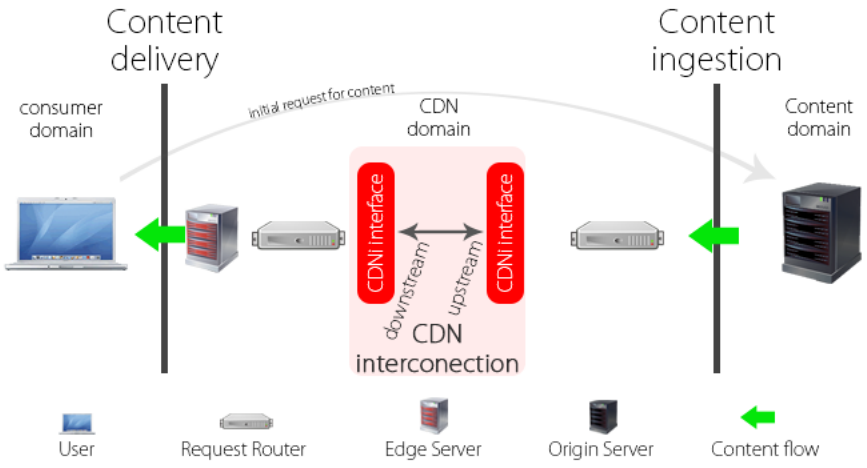

Figure 25: *Show relationships between CDN provideres, their functions and concept of domains in use case with live stream from previous scope*

3.3.2 Requirements and Capabilities

In this section we focus on requirements of CDNi and their capabilities which shall have. Capabilities incurred from requirements in ETSI TS 102 990 [22] and then will be more specified in ETSI TS 182 032 [21]. Capabilities are switched to main categories. One category is mandatory capabilities of CDNi and second category is optional capabilities.

Mandatory capabilities

First mandatory capability set is Interconnection control. Interconnection control cares and manages connectivity between two CDNs. Second is Request Routing which has to be responsible for redirection of clients requests directly to the Edge server of a Downstream CDN based on for example geological location or extraction of Edge server. Third mandatory capability is Content distribution from an Upstream CDN to a Downstream CDN. Fourth capability is Footprint exchange. Footprint exchange is necessary capability for request routing. Upstream CDN have to know which footprints a Downstream can serve. Fifth mandatory capability is Metadata exchange that contains content related information. Sixth is Content Status exchange that is used to exchange real-time status related to the content. Seventh is Report exchange that will be useful for request routing. Based on reports from a Downstream CDN we can make decision about routing requests to Edge servers. The last one mandatory capability is Capability exchange which mediates information of capability from a Downstream CDN to Upstream CDN. [21]

Optional capabilities

These capabilities in this master thesis are not important, because our implementation is based on the mandatory capabilities and optional capabilities only for informative character. Therefore we gave only hits. Optional capabilities are: Metadata Defined Reporting, Content Integrity Control, Content Adaptation, Multi-Segment Content Support, Content Access Control, Content Security and DRM, Custom Capability Support.

3.4 CDNi structure

This section will be focused on description of structure of CDNi. We specify CDN adapter as two interfaces with graphical user interface to control it. CDNi will be attached to CDN as new node. These nodes are web servers in the real world. We will describe it in the latter chapter. As we can see on Figure 26. CDNi contains two parts which are CDNi interface manager and CDNi interface.

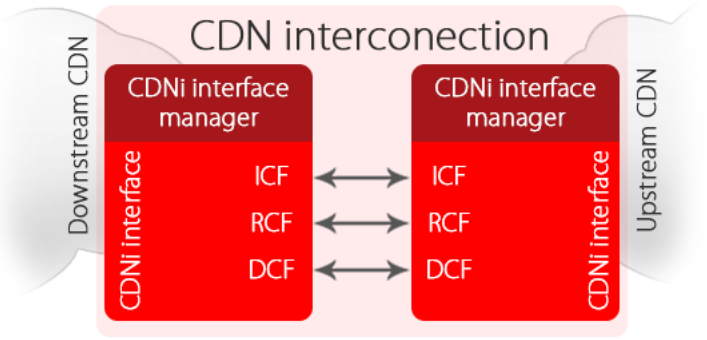

Figure 26: *CDNi structure*

3.4.1 CDNi interface

The main part of CDNi structure is CDNi interface. As the document ETSI TS 182 032 [21] describes, CDNi interface has three different level of functionalities. First level takes care about content delivery. Second level takes care about controlling of content and requests. Third level takes care about interconnections itself. As we can see on Figure 26. every level has own abbreviation. CDN Interconnection Control Function (ICF) contains Footprint exchange, Capability Exchange, Network Status Reporting and Network Logging. Request and Content Control Function (RCF) is responsible for Metadata exchange function, Content request function and Content status reporting. Distribution of Content Function (DCF) is responsible for Transfer of file-based content, Publication and streaming of stream-based content and Content metadata distribution.

3.4.2 CDNi interface manager

CDNi interface manager is priority designed for administrators. Interface manager allows administrators to create, manage, delete and control interconnections between CDN, configuration of properties of CDN or to monitor the real time status of CDN based on real time logs. Interface manager is necessary or suitable for several reasons. Provider of content delivery network has interconnection under control and can maintain it. Interconnection will be created only if providers of CDNs will make a deal. There can occur a situation where content provider doesn't want to share content from different CDNs.

3.5 CDNi basic life cycle

This section describe basic life cycle of CDN. Basic life cycle is a representation on Figure 27 based on ETSI TS 182 032 [21]. Cycle starts after making a deal between CDN providers. CDN providers can make deal by using interconnection manage. This deal requires attention of both providers because one of them has to offer deal and the other has to accept a deal. After accepted deal next processes are fully automatic. Upstream and downstream CDN can start process of establishing interconnection. After this point both CDNs know about of theirs peers and can start exchange theirs footprints and capabilities. Footprints and capabilities are necessary for content distribution and routing request. After these five procedures, CDNi is ready to deliver content. Next five procedures can be repeated and their order may be changed. But some of procedures are based on Upstream or Downstream initiated content distribution. After this procedure there may occur Content exchange, Single delivery request, initiated content deletion or initiated reporting.
These procedure demonstrate main functionality of CDNi. At next section we focus on implementing it.

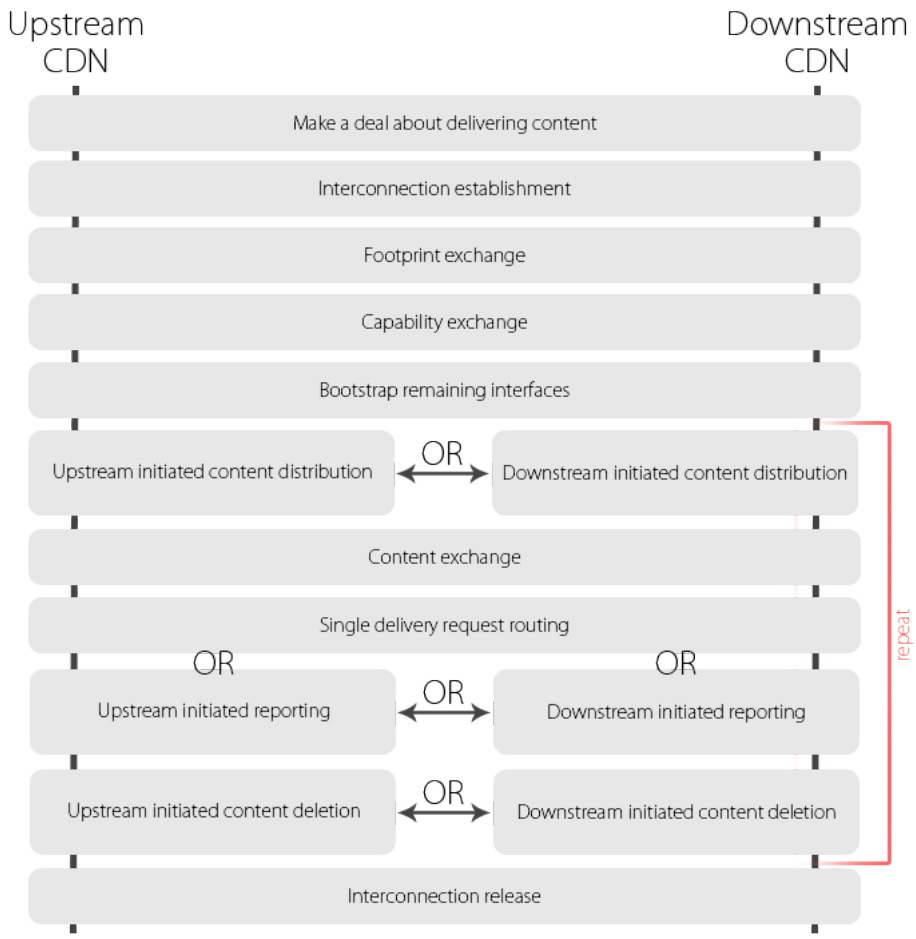

Figure 27: *CDNi basic life cycle*

3.6 CDN structure

Our network for distribution content will consists of two CDNs. One of them will be CISCO proprietary CDN (CISCO Internet Streamer - CDS) [25] and second one will be Dummy CDN. Dummy CDN will simulate behavior of the real CDN. Dummy CDN will be our implementation. Both of CDNs will have CDN interface. Every web server will represent CDNi interface with CDNi manager as it is shown on Figure 26.

3.7 Verification of solution

We can verify our solution using streaming of video content from origin server to customers. It can be seen on Figure 28. Red shapes show us stream flow from origin server to customers. Every customer will be in different footprint (different network). Every CDN will be able to serve content to different footprint. CDNs will be connected using our CDN interconnection solution. User devices will use our application for watch video content.

Each user will start to watch the same video stream using our application. We will be able to see who is sender of video content to user devices using Wireshark software.

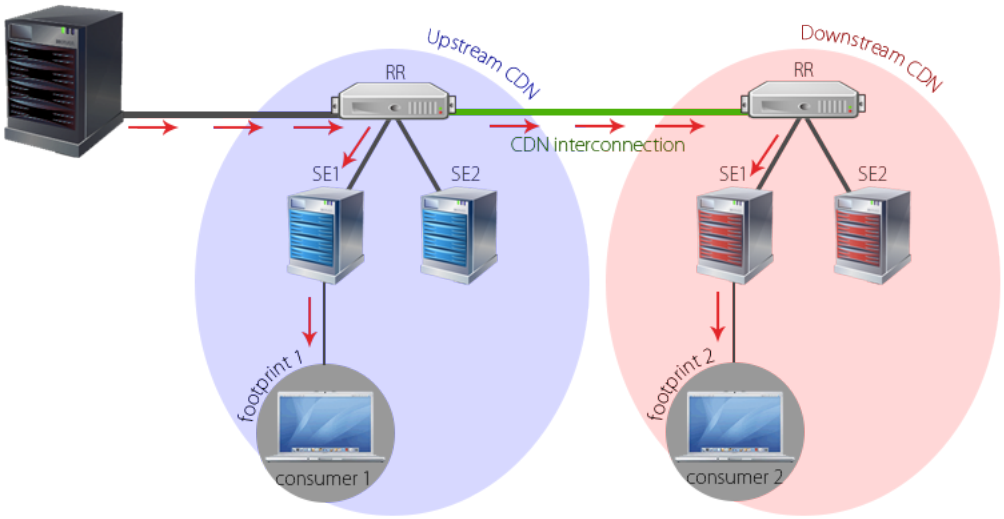

Figure 28: *Show topology for verification of solution*

4 Implementation

4.1 Implementation tools choice

4.1.1 Origin server

Origin server was described in subsection 3.1.3. In this part of system design, our requirements on Origin server do not change. The server only needs to support HTTP protocol for on demand video. For transferring of files to the server, we can use FTP protocol. Our preferred type of streaming is HTTP Live Stream (HLS) by Apple, which is supported by all of the most common devices. If we wanted to provide a live streaming of content, we would require a specialised software. The software for live streaming of media content to a network may be VLC media player [24] or Wowza [23] media system.

4.1.2 CDNi interface

As we indicated in subsection 3.6, CDNi interface will be a web server application. We decided to implement CDNi interface in PHP programming language with extensions for PostgreSQL or MySQL database support. Our decision is based on years of personal experience with PHP. We would like to use PHP framework that helps us to develop faster. PHP frameworks have more advantages. Source code is consistent using Model View Controller (MVC) and we can write queries independent on the database used.

There are many frameworks for PHP (Zend Framework 2, CakePHP, Nette, CodeIgniter, Yii, etc.). We have chosen the **Yii** [26] PHP framework. This framework is relatively new. Yii is under development since 2008 and the name is an abbreviation for "Yes It Is". Yii is one of the robust frameworks, however it is very fast and has a good caching system. The database layer of Yii is based on active record pattern and lazy loading. Thanks to this layer, written queries are no longer dependant on the database used. One of the useful features is a code generator. Yii can generate a code (model, controller and views) from the structure of database table. [27]

CDNi interface manager will be implemented as a web application by using Yii framework, **Javascipt** and **AJAX** technologies. Javascipt and AJAX are well known web technologies used to create an interactive interface.

We have to implement a communication between our CDNi interfaces. To create a communication, we use **Simple Object Access Protocol** (SOAP) [29] protocol. SOAP is a protocol for transferring of messages between two points. The messages are based on the XML technologies. The main advantage of using the SOAP is a HTTP protocol used to transfer the data. Next advantage of SOAP is its independance on platform. Few extensions are available for Yii framework to control SOAP communication [28]. SOAP can use Web Services Description Language (WSDL) that abstractly describes network services. In a chapter later in this work, we show our implemented SOAP messages.

4.1.3 Mobile application

User interface as a mobile application we described in subsection 3.1.2 becomes less important in this stage of our project. But still, the implementation of an application will be on Android platform.

4.1.4 Media tools

We need media tools for segmentation and encoding of the media content and preparation for HLS streaming. We have prepared three tools for this task. First one is the VLC Media Player [24]. VLC Media Player is a multiplatform application. Linux and Apple OS X are the platforms we are interested in. We have a Debian distribution of Linux on the Origin server and OS X is the main platform for development. VLC is capable of encoding, segmenting and streaming of the media content. VLC supports command line tools as well. Example 1. demonstrates a VLC command. This command creates segments of media content with .m3u8 playlist file of HLS stream.

Example 1: *Encode and segmentation by VLC*

```
vlc -I dummy /var/myvideos/video.mpg vlc://quit --sout='#transcode{width=320,height=240,↩
  » fps=25,vcodec=h264,vb=256,venc=x264{aud,profile=baseline,level=30,keyint=30,ref=1},↩
  » acodec=mp3,ab=96}:std{access=livehttp{seglen=10,delsegs=false,numsegs=0,index=/var/↩
  » www/streaming/mystream.m3u8,index-url=http://mydomain.com/streaming/mystream↩
  » -########.ts},mux=ts{use-key-frames},dst=/var/www/streaming/mystream-########.ts}'
```

Apple provides the tools for creating of segments from media content and for creating variants of playlists. These tools are available only for Apple developers with active account. Mediafilesegmenter and Variantplaylistcreator are two interesting software products from Apple. Mediafilesegmenter helps us to create segments from encoded files with .m3u8 playlist file and variant .plist file. It is demostrated on the Example 2. Variantplaylistcreator helps

us to create single .m3u8 playlist from several .m3u8 playlists combined with .plist files that contain different resolutions of the media content. It is demonstrated on the Example 3.

Example 2: *Segmentation using Mediafilesegmenter*

```
/usr/bin/mediafilesegmenter -I -f mymedia_hi -f mymedia_hi.mp4
```

Example 3: *Creation variable play list using Variantplaylistcreator*

```
/usr/bin/variantplaylistcreator -o mymedia_all.m3u8 http://mywebserver/mymedia_lo/↩
  » prog_index.m3u8 mymedia_lo.plist http://mywebserver/mymedia_med/prog_index.m3u8 ↩
  » mymedia_med.plist http://mywebserver/mymedia_hi/prog_index.m3u8 mymedia_hi.plist
```

Apple tools has one disadvantage for us. They don't have a tool for encoding of a video into different resolutions. We found one open source video transcoder called HandBrake. It has an intuitive graphical interface used to set required output format. HandBrake capabilities are sufficient for our needs.

4.2 Prototype of mobile application

The prototype was created as a mobile application based on web technologies. We used the framework called PhoneGap (Cordova). The framework works with many platforms such as Android, iOS, BlackBerry, etc. This framework adds web objects to project with the possibility to control device interfaces. This feature allows the creating of mobile applications just by using HTML5 and JavaScript. HTML version 5 contains the video tag, which we use for the playback of a video (more about HTML 5, video tag and supported files can be found in the Samsung Smart TV chapter). We created graphical user interface using jQuery mobile, the extension for JavaScript. This prototype will be used for comparison of delivering a multimedia content to a device by using different technologies.

Prototype contains only one Origin server and a mobile application. User can watch movies using the application, rate movies and search movie based on genre and name.

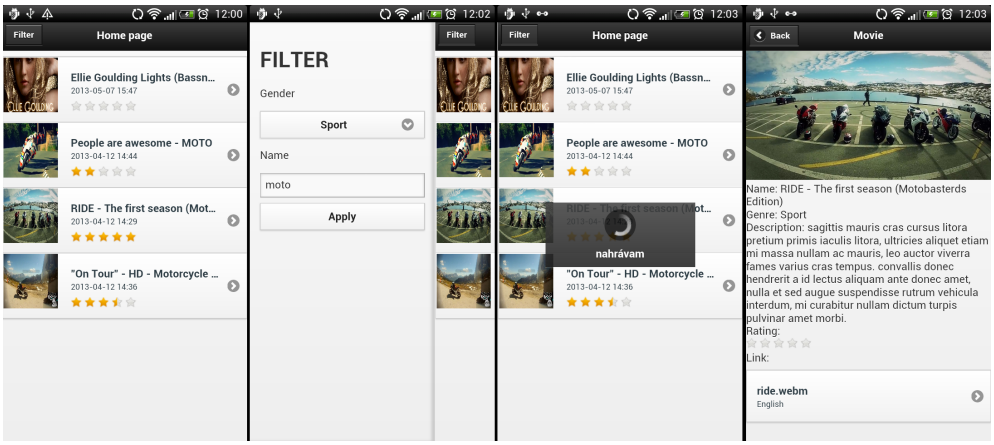

Figure 29: *Screenshots of prototype application (list movies, filter, loading, movie info)*

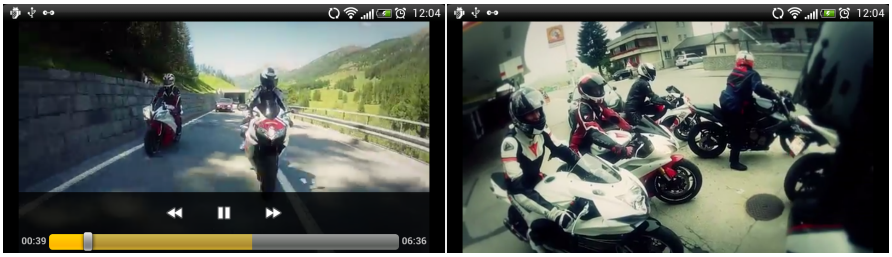

Figure 30: *Screenshots of playing movie at prototype application*

4.3 Generating of test content

As we described at subsection 4.1.4, we used Apple tools to create HLS playlists and HandBrake to encode video content. We used video content that we made. We encoded original content into four different resolution and quality levels (low, medium, high, xhigh) using HandBrake. We did the same steps for every resolution to create the HLS segments. Segments were created using Mediafilesegmenter with options "-I" (create variant .plist), "-i" (name of .m3u8 file) and "-b" (url to content). We used the option "-I" to create variable HLS playlist.

Example 4: *Commands for generating test content*

```
Richards-MacBook-Air:high richardrostecky$ /usr/bin/mediafilesegmenter -b http://www.↩
   » closerit.com/hls/high -I -i mototrip2013.m3u8 ./MotoTrip2013.mp4
```

```
Richards-MacBook-Air:xhigh richardrostecky$ /usr/bin/mediafilesegmenter -b http://www.↩
   » closerit.com/hls/xhigh -I -i mototrip2013.m3u8 ./MotoTrip2013.mp4
Richards-MacBook-Air:medium richardrostecky$ /usr/bin/mediafilesegmenter -b http://www.↩
   » closerit.com/hls/medium -I -i mototrip2013.m3u8 ./MotoTrip2013.mp4
Richards-MacBook-Air:low richardrostecky$ /usr/bin/mediafilesegmenter -b http://www.↩
   » closerit.com/hls/low -I -i mototrip2013.m3u8 ./MotoTrip2013.mp4
Richards-MacBook-Air:hls richardrostecky$ /usr/bin/variantplaylistcreator -o ↩
   » mototrip2013all.m3u8 -resolution-tags ./xhight/mototrip2013.m3u8 ./xhight/↩
   » MotoTrip2013.plist ./high/mototrip2013.m3u8 ./high/MotoTrip2013.plist ./medium/↩
   » mototrip2013.m3u8 ./medium/MotoTrip2013.plist ./low/mototrip2013.m3u8 ./low/↩
   » MotoTrip2013.plist
```

4.4 Set up CDN

Cisco Internet Streamer offers setup through graphical user interface. GUI has a lot of features. We set mandatory settings for streaming of our content using CDN. First, we had to create a new Content origin. It is important to set name, Origin server URL and Service routing domain name. In our case, name was "closerit", url was "www.closerit.com" and domain name was "rfqdn3.cdn.ab.sk". Next, we were able to create new Delivery service. Delivery service view is shown on the Figure 31.

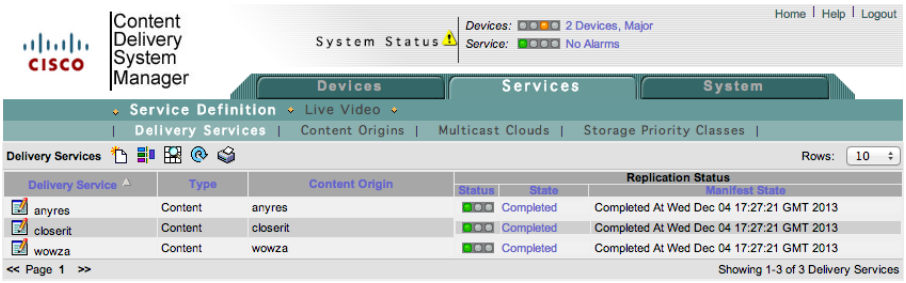

Figure 31: *Screenshot GUI Cisco Internet Streamer*

Delivery service contain two main fields to set, except the name. The first is a content origin that we created in first step and the second is a Preposition storage quota. We set quota on 50000 MB. Next, we had to assign Service engine (Edge servers) that will deliver content to end users. Table of Service engines is shown on the Figure 32. The last change that we had to do was to change a location of our content on origin server. We changed the URL from "http://www.closerit.com/hls-android/fileSequenceXX.ts" to "http://rfqdn3.cdn.ab.sk/hls-android/fileSequenceXX.ts" for every segments in mototrip2013.m3u8 file. We checked these settings by playing the stream in media player and using Wireshark to capture packets. Packets contained IP address of Service engine instead of IP address of the origin server.

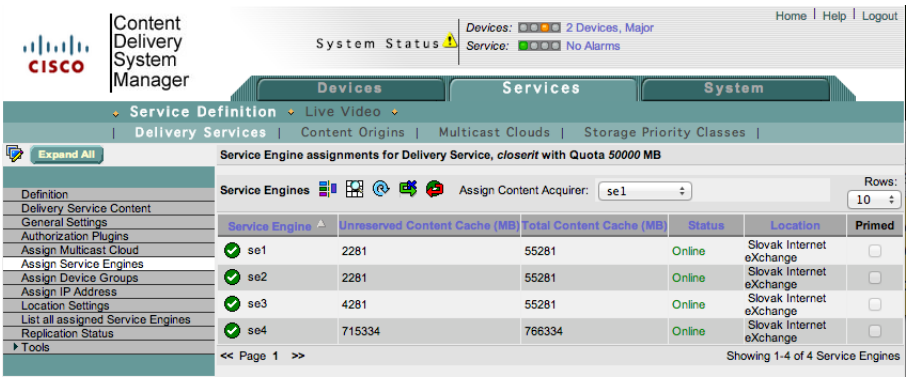

Figure 32: *Screenshot GUI Cisco Internet Streamer*

4.5 CDNi interface

As described above, CDNi consists of a several pieces that care about the main functionality of CDN interconnection. In the implementation, we have focused on mandatory functionality that helped us to make test scenario described in the verification section. We followed principles of object oriented programming and model view controller that makes application more comprendious and ready for future upgrade, for example: implementation API of other content delivery networks.

4.5.1 Database

In the end, database was created in the MySQL. Role of a database is to store the information about interconnection and very a important role is to distinguish different types of CDN interface. Different types of CDN interface are distinguished using the attribute stored in a database and have different behaviors of CDNi. Different CDN interfaces control different types of CDNs with the same PHP application code even for different behaviors. CDN interface is the same on every gateway. Gateway is a server where our application runs and communicates with CDN. You can see the structure of database on Figure 33.

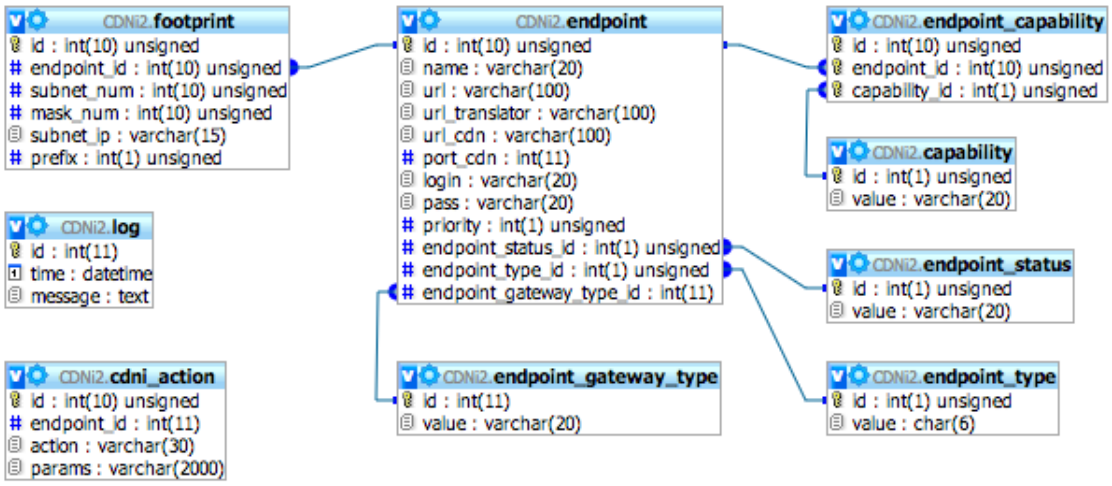

Figure 33: *Final database structure of CDN interface*

The main table of database is the `endpoint` table. Endpoint table represents endpoints of interconnection. Endpoint table contains foreign key to `endpoint_gateway_type` table that indicates which endpoint of interconnection is local and remote endpoint. Local endpoint is the endpoint controlling local CDN. Remote endpoint is an endpoint of remote CDN interface. Graphical view of endpoint logic is shown on Figure 34.

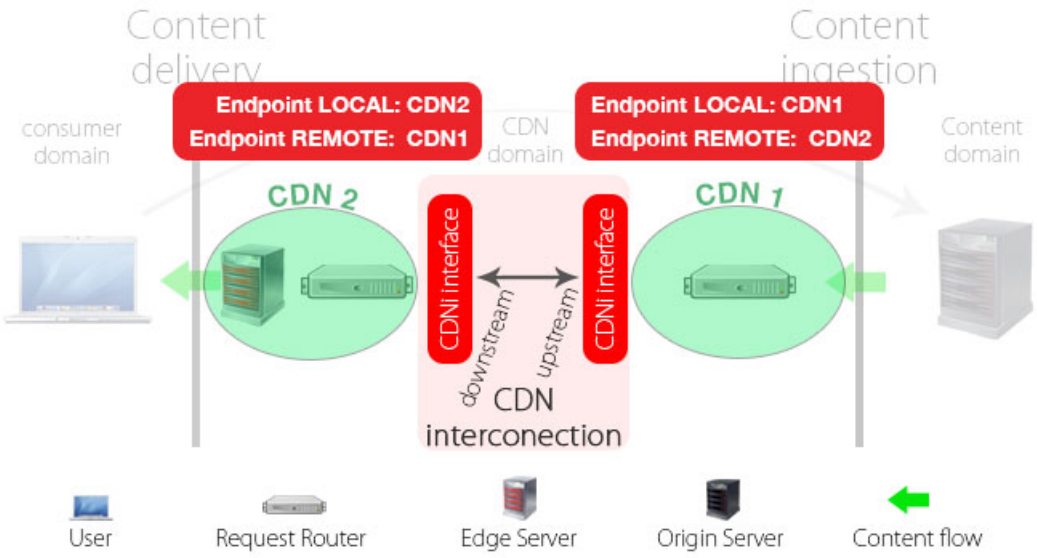

Figure 34: *Describe end point logic.*

Table `endpoint` contains some more attributes to describe CDNi interconnection like the URL with the port of content delivery network where we can connect to the API, login and password to authenticate to a CDN. Also it contains foreign key to `endpoint_type` table that identifies type of endpoint. A savoir type of CDNi interface. In our case of implementation we can differentiate two types of CDN. First one is CISCO Internet Streamer CDS and second one is Dummy CDN. Dummy CDN imitates real CDN. Its purpose is only to test and verify our implementation. The next one foreign key in `endpoint` table is foreign key point for `footprint` table where are stored all footprints of endpoint. The last one foreign key points to `endpoint_capability` table. This table stores extensions of files that CDN can handle.

Table `cdni_action` stores all the actions that CDN interface do with a CDN. It also stores the information about the exchange of Content origins between CDN. We will explain it later. The last table `log` stores all of the CDNi logging messages that contain information about actions done.

4.5.2 Application

Structure of our application is derived from YII application structure. These directories are important for us:

Example 5: *Important directories in application*

```
/protected/config           //include configuration files of application
/protected/controllers      //include controllers
/protected/models           //include base models and models that extend base models
/protected/views            //include views belong controllers
```

YII framework with GIIX generator extension helps us to create base models and CRUDs for every table in database. We used the GIIX generator to create separate base database model and extended model and for much better handling of many-many and one-many database relationships. Base model is very helpful in the case of future update. All changes that we make and all the functionality we add is done in the extended model. In the case of future database changes, we only generate the base model and keep our changes in the extended model. As we can see on Figure 35, GIIX generator generated two files for `endpoint` table. One of them is stored right in the directory with models. It is extended model. The second one is the base model and is stored in `/_base` subdirectory under the models directory. GIIX crud generator created more files. One of these files is the controller and the others are views.

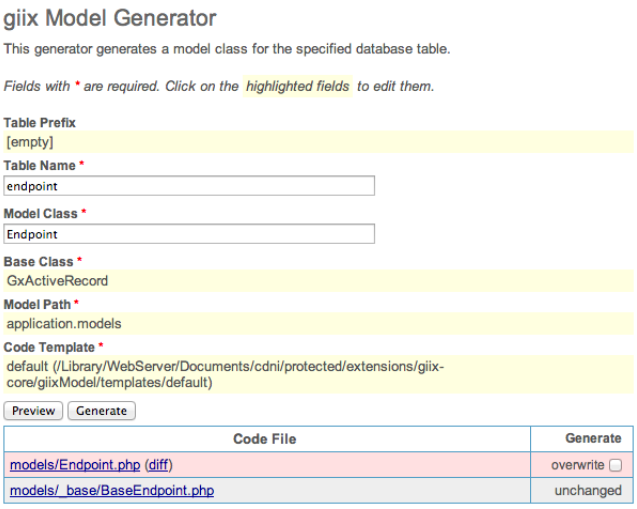

Figure 35: *Yii framework with Giix extension for generate Models.*

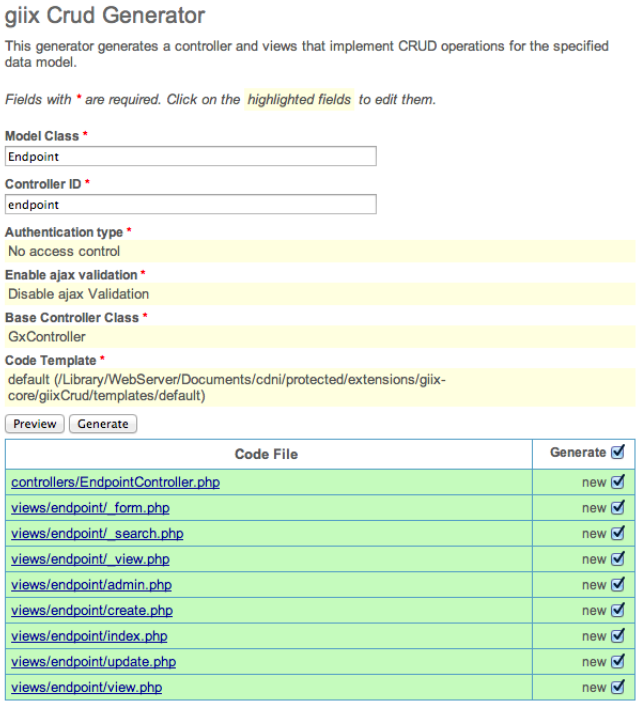

Figure 36: *Yii framework with Giix extension to create CRUD views and controllers.*

Except the files that were generated by GIIX extension of Yii framework, we had to create a few more controllers and models. As we have written above, our implementation also contain

Dummy CDN. Dummy CDN is the part of CDN interface. `DummyCdnController` contain basic logic and routing of Dummy CDN. In our database, we have the table `endpoint`, but we differentiate two types of endpoints - Local and Remote. This is the reason why we have created two controllers, `EndpointLocalController` and `EndpointRemoteController`. `EndpointLocalController` takes care about setting up of local endpoint attributes that were described above. `EndpointRemoteController` control the logic of all interconnection operations. That means that the controller is able to create an offer for the remote endpoint, exchange all the necessary attributes, accept received offer and delete interconnection. All these functions used to communicate between two CDN interface are exchanged over the SOAP protocol. SOAP functionality need its own controller. Therefore we create `SoapController`. Also, we created special controller for Cisco CDN called `UrlTranslationServiceController`. Role of this controller will be explained later. The last controller is the `CronController`. This controller is ready for the implementation of periodical updates between CDNi interfaces. Models in our application contain one more important file. This file contain implementation of the Cisco Internet Streamer CDS 3.2 API. Our application is designed to allow future upgrades. New CDN needs to add own implemented API files for models and add necessary operation to switch statements in `EndpointLocalModel`, `EndpointRemoteController` and `SoapController`.

4.5.3 Life cycle of application

On the Figure 37 is shown life cycle of our application. The life cycle starts with the creating of interconnections that have to be initiated by administrator of downstream CDN. Administrator creates an offer that will be send to upstream CDN using the graphical user interface. This functionality is handled by function `actionCreate()` located in `endpointRemoteController`. Every interconnection is defined by the name of an endpoint and a URL to endpoint. Function `actionCreate()` calls three function on the remote CDN interface using the web service. First function called is the `setOffer()`. It creates a new entry in the `endpoint` table. The second function called is `setFootprint()`. This function set downstream's footprints on upstream CDN interface associated with the new endpoint entry. Third function of this iteration is `setCapabilities()`. As the name indicates, it sets downstream CDN's capabilities on upstream CDN interface associated with new endpoint entry.

After sending of the offer to create an interconnection form downstream CDN to upstream CDN, administrator of upstream CDN can see received offer between the interconnections. To successfully create an interconnection, he has to accept the offer. If an administration accepts the offer, `actionAcceptOffer()` function locate in `endpointRemoteController` is called. This function calls functions on downstream CDN used by web service. First function called is the `acceptOffer()`. This function only changes the state of interconnection on downstream CDN. Second function called is `setFootprints()` and the third is `setCapabilities()`. These functions are the same as we describe above, but in this case, the footprints and capabilities are set up on downstream CDN interface. Function `actionAcceptOffer()` contain one important function that get local Content origin. Name of this function is `getListContentOrigin`. In the case that the local endpoint type is Cisco, Content origin comes directly from the Cisco CDN. Otherwise, if the type of local endpoint is Dummy, local Content origins are stored in associative array in `cdni_action` table. The number of Content origins define how many times we will call the next functions. What functions will be next depends on the type of remote CDN. If the type of remote CDN is Dummy, the next functions will be `getListFqdn()` and `createDummyContentOrigin()`. Function `getListFqdn()` returns the array of URLs that can be associated with Content origins. This URLs are used to identify which origin server contain requested content. Every CDN has to define possible FQDN. If CDN doesn't have free FQDN, content origin will not be created. Function `createDummyContentOrigin()` creates associative array that contains origin server URL mapped to FQDN URLs. This associative array helps Dummy CDN handle requests from the hosts. If the type of remote CDN is Cisco, than next functions are `getListFqdn()`, `createContentOrigin()`, `createDeliveryService()`, `getDevices()` and `assignSEs()`. First function is already described above. The second function, `createContentOrigin()` creates content origin in Cisco CDS used by Cisco API and return ID of the content origin that will be used by second function `createDeliveryService()`. This function returns ID of created delivery service. Delivery servicse need to be assigned to service engines that will handle requests from users. Therefore, function `getDevices()` that return list of devices is called first. These devices are used in the last function, `assignSEs()`. This function finally assigns service engines to delivery service. In the end, last two function are called in both cases. One function is `getRemoteAssociateOriginTofqdn()` and second is `setRemoteAssociateOriginTofqdn()`. These functions update list of remote Content origins with their FQDN. This was the last operation necessary to create interconnection

between two CDN. Now, downstream CDN is prepared to handle request from upstream CDN.

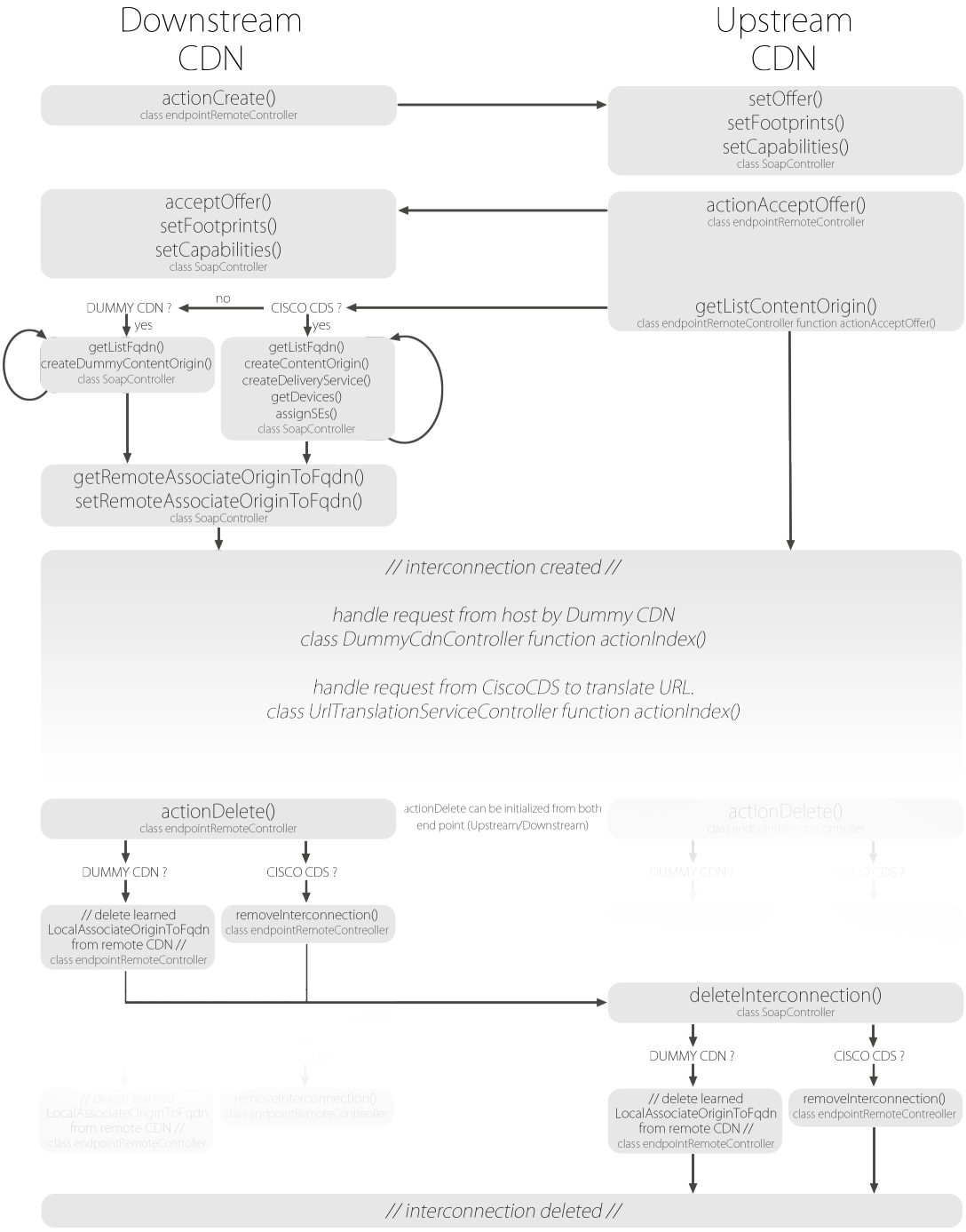

Figure 37: *Life cycle of CDN interface.*

Main role of the content delivery network is to handle user request and make decisions about what edge server will deliver the requested content. Our CDN interface has to know how to control the routing request. Dummy CDN is our implementation than controls routing we implemented directly in the logic of request handling. Cisco CDS has its own logic apart from our CDN interconnection. Our first idea was to create selector files that Cisco CDS use on the top of request routing engine work flow. This file contain a list of CDNs to choose from using geo location. This method is implemented too, but problem is in incoming requests from upstream Cisco CDS to downstream CDN. These requests only contain path to content but we couldn't distinguish who the origin server and the owner of content is. Therefore we couldn't resolve the request. Then the idea with the last-resort translator came. Last resort occurs if Cisco CDS doesn't explicitly match subnet in the coverage zone file. Last resort has to be set up and allowed in Cisco CDS. The IP address and communication port of third-party translator server is set. Web services are used for the communication. We implemented this translator server in our CDN interface. We created virtual host on Apache server where we run the CDN interface and we used module rewrite to translate redirect requests from Cisco CDS to our CDN interface application. Requests contain XML with URL, client IP address and flag signalling that the URL is signed. From URL is possible to parse the type of content, FQDN, protocol used and path to the content. We use FQDN to find association between FQDN and origin server in the array `LocalAssociateOriginToFqdn` stored in `cdni_action` table. If we have an origin server, we can find matching footprints between our interconnections and find origin server on remote CDN interface that matches with our origin server from incoming request. If we find this match, we can make decision and response to Cisco CDS with new URL, where Cisco can redirect requests to. If we don't have any match in interconnections, our response have to contain URL address of service engine belonging to Cisco CDS that generate request. Our purpose of this decision is to handle every request. As we mentioned above, coverage zone file contains footprints that Cisco CDS can handle. This coverage zone file is created and registered to Cisco CDS in the case, when the administrator change local footprints in the CDN interface. Coverage zone file is assigned globally to all service engines. In the case of Dummy CDN, its routing request logic is very similar, but Dummy CDN communicate directly with end user and therefore response is HTTP message with the status code 302 - moved temporary with new URL.

Administrator of CDN interface is able to delete CDNi interconnection using graphical user interface. Function `actionDelete()` located in `endpointRemote` class initiates deletion of interconnection on local endpoint and remote endpoint. If the CDN type of local

endpoint is Dummy, it is only necessary to remove learned items from created interconnection from `LocalAssociateOriginToFqdn`. If local endpoint type is Cisco than is necessary do more operations to delete an interconnection. As we mentioned above, if Cisco is downstream CDN, we have to create the interconnection for Content origin, Delivery service and assigned Service engines to delivery service. With every of these actions, we made an entry to the `cdni_action` table. Based on these entries, we know what we have done with Cisco CDS and what we have to revert to initial state. After these deletions on local endpoint, `actionDelete()` call the function `deleteInterconnection()` located in `SoapController` on the remote endpoint used by web services. The function initiates the same procedures as on the local endpoint.

5 Testing

We mentioned in section System design how we will verify our solution. Based on it, we create topology that is shown on Figure 38. We had available two Cisco CDS and one our Dummy CDN. First Cisco CDS is located in Bratislava in Slovakia and second Cisco CDS is located in Netherlands. Dummy CDN is a part of CDN interface. All CDN interfaces and origin servers are located in Bratislava. Dashed line on figure indicates one physical device.

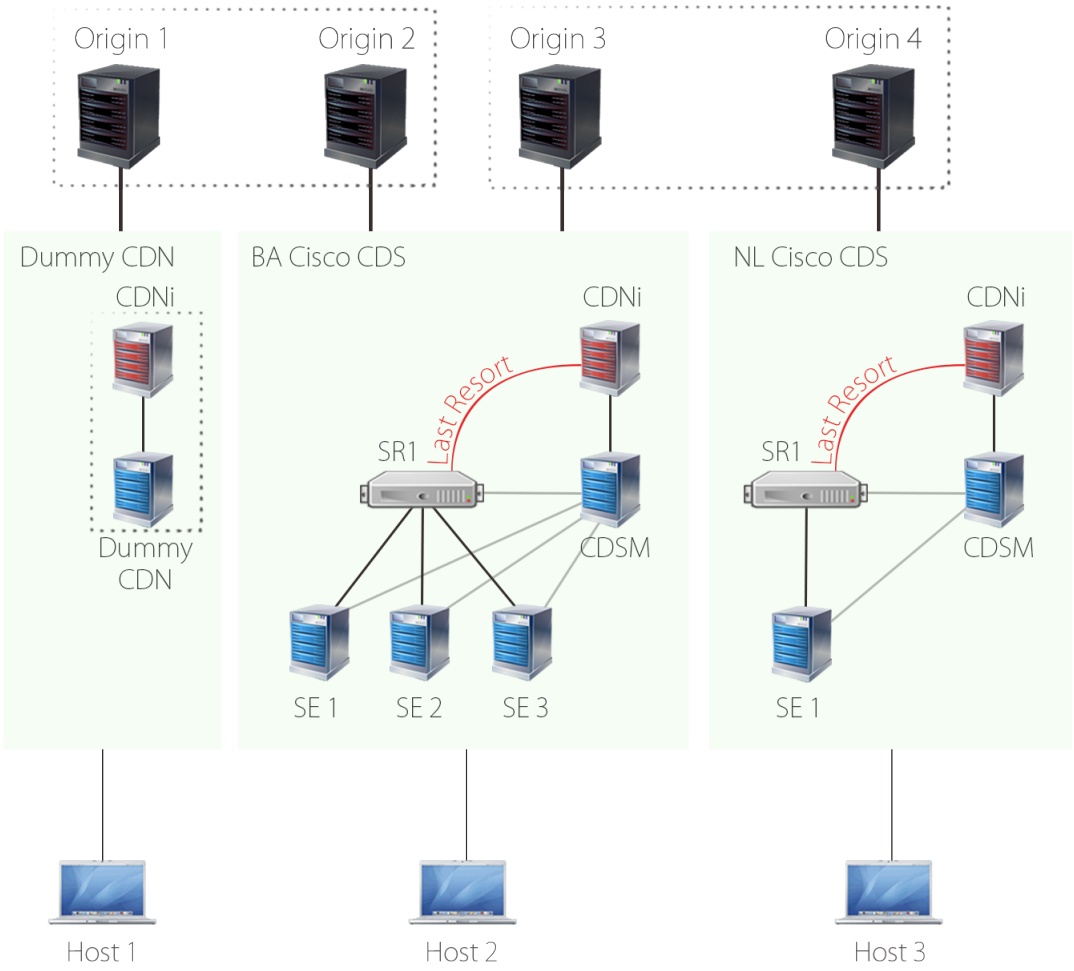

Figure 38: *Topology for testing CDNi.*

On Figure 39 is shown table of used devices. Verification of solutions has to demonstrate functionality of CDN interface. We made set of tests together with doing verification. We want to know how will be increasing delay when will be URL translator used.

Devices		
CDN	**NAME**	**IP ADDRESS/DOMAIN NAME**
DUMMY CDN	CDNi	147.175.15.42
	Dummy CDN	147.175.15.42
	Origin 1	147.175.15.43
		origin1.rostecky.sk
BA CISCO CDS	CDNi	147.175.15.41
	CDSM	147.175.204.100
	SR1	147.175.204.110
	SE1	147.175.204.120
	SE2	147.175.204.121
	SE3	147.175.204.122
	Origin 2	147.175.15.43
		origin2.rostecky.sk
	Origin 3	147.175.15.94
		origin3.rostecky.sk
NL CISCO CDS	CDNi	147.175.15.90
	CDSM	139.63.240.10
	SR	139.63.240.12
	SE	139.63.240.11
	Origin 4	147.175.15.94
		origin4.rostecky.sk

Figure 39: *List of devices used in tests.*

We prepared testing content in origin servers. Content was same on every origin server. Every CDN has assigned Origin server as it is shown on Figure 38. We created interconnection between all three CDNs. In first test, Cisco CDS in Bratislava was upstream CDN and the other CDNs were downstream CDNs. Our testing hosts were assigned to CDNs based on footprints. Every host was in different CDN and they load same web page. In first test was used a web page store on origin2.rostecky.sk. Successful load of web page indicates shown picture on it. Picture will not be loaded when host will not be handled by one of CDNs. It is shown on Figure 40. Under image is link to m3u8 file of test multimedia content. We can open it in VLC player or Quick Time player. When video is played we know that host was handled by some CDN. We can determine which CDN handles user's request based on sniffed communication or we can found logs about translation URL in CDNi user interface.

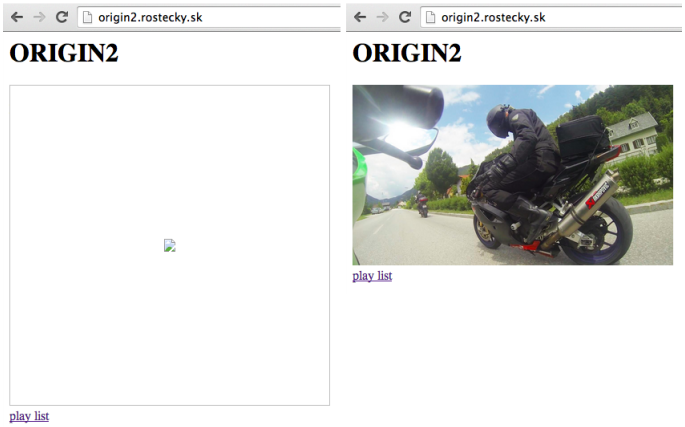

Figure 40: *Demonstration of unsucesfull and sucesfull loading of origin2.*

Second test is same like first but upstream CDN was Cisco CDS in Netherlands. These tests shown us function of CDN interface. But we want to know delay too. We sniffed all communication from user when user made request to content. Before request, we cleared cache of DNS on host's PC. We analyzed sniffed communication and created tables and graphs according to results.

First three graphs with tables are information graphs that show us order of requests from user and their delays. Fourth and fifth graphs contain average needful delay to the first request directly to content. There are compare delays belongs to hosts in upstream CDN with hosts belongs to downstream CDN. There are two types of delays. One of them is with DNS request and second one is without DNS requests. Sixth graph show us request's delays on URL translator on both Cisco CDS. The last one graph show us average delays of needful requests. There are compared normal request with result that contains 302 redirect, request that contains URL translation with result 302 redirect and request to resolve DNS.

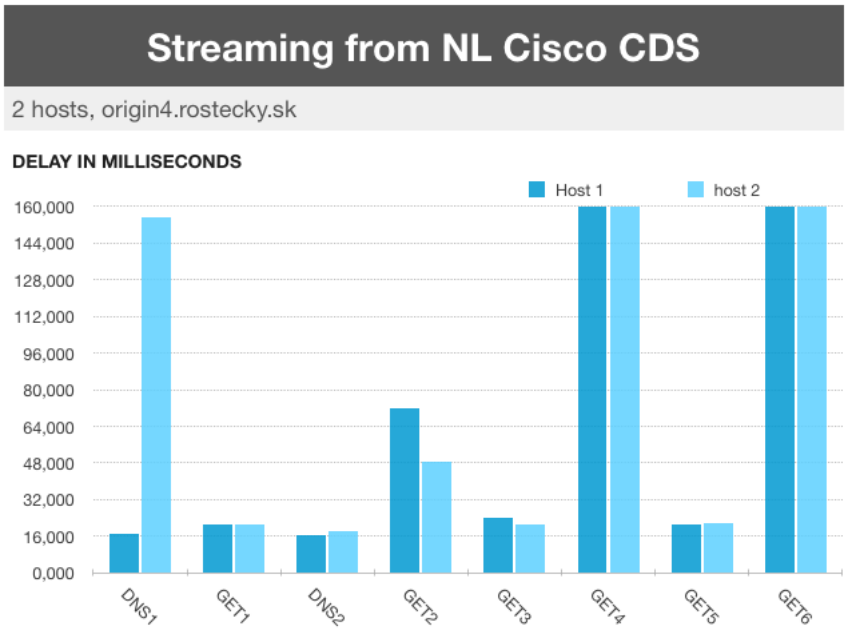

Figure 41: *Graph of requests needed to reached content on Cisco CDS in Netherland.*

	HOST 1	HOST 2	REQUESTS
DNS1	16,740	155,017	Standard query A rfqdn4.cdn.rostecky.sk
GET1	21,314	21,387	**GET** http://rfqdn4.cdn.rostecky.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1 **response**: HTTP/1.1 302 Moved Temporarily
DNS2	16,158	18,166	Standard query A se1.se.rfqdn4.cdn.rostecky.sk
GET2	71,915	48,411	**GET** http://se1.se.rfqdn4.cdn.rostecky.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1 **response**: HTTP/1.1 206 Partial Content (text/plain)
GET3	23,768	21,393	**GET** http://rfqdn4.cdn.rostecky.sk/motoTrip/fileSequence0.ts HTTP/1.1 **response**: HTTP/1.1 302 Moved Temporarily
GET4	10073,427	10158,501	**GET** http://se1.se.rfqdn4.cdn.rostecky.sk/motoTrip/fileSequence0.ts HTTP/1.1 **response**: HTTP/1.1 206 Partial Content (text/texmacs)
GET5	21,031	21,493	**GET** http://rfqdn4.cdn.rostecky.sk/motoTrip/fileSequence1.ts HTTP/1.1 **response**: HTTP/1.1 302 Moved Temporarily
GET6	6653,887	6708,877	**GET** http://se1.se.rfqdn4.cdn.rostecky.sk/motoTrip/fileSequence1.ts HTTP/1.1 **response**: HTTP/1.1 206 Partial Content (text/texmacs)

Figure 42: *Table of requests to Cisco CDS in Netherland.*

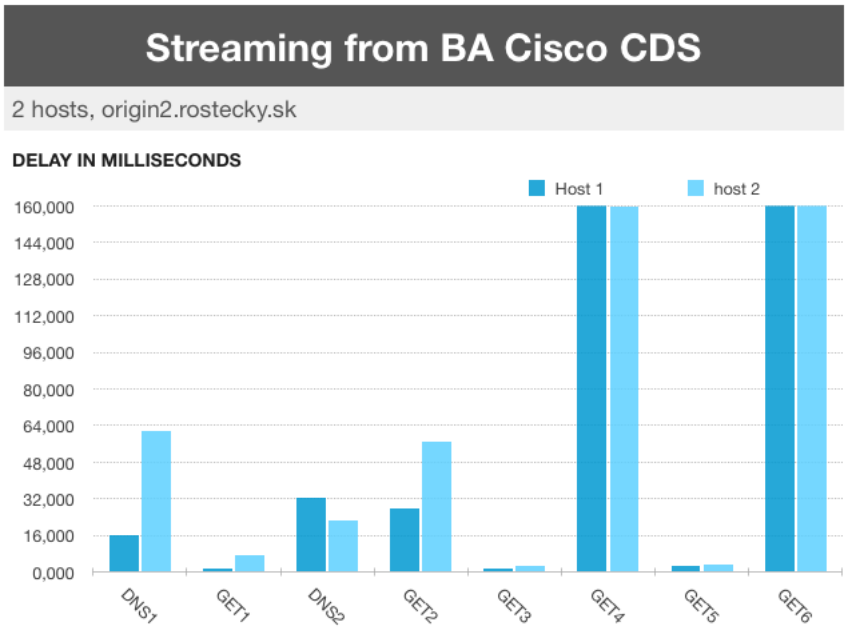

Figure 43: *Graph of requests needed to reached content on Cisco CDS in Bratislava.*

	HOST 1	HOST 2	REQUESTS
DNS1	16,232	61,599	Standard query A rfqdn3.cdn.ab.sk
GET1	1,691	7,178	**GET** http://rfqdn3.cdn.ab.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1 **response**: HTTP/1.1 302 Moved Temporarily
DNS2	32,587	22,535	Standard query A se3.se.rfqdn3.cdn.ab.sk
GET2	27,695	57,240	**GET** http://se3.se.rfqdn3.cdn.ab.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1 **response**: HTTP/1.1 206 Partial Content (text/plain)
GET3	1,523	2,722	**GET** http://rfqdn4.cdn.rostecky.sk/motoTrip/fileSequence0.ts HTTP/1.1 **response**: HTTP/1.1 302 Moved Temporarily
GET4	10259,893	78497,270	**GET** http://se3.se.rfqdn3.cdn.ab.sk/motoTrip/fileSequence0.ts HTTP/1.1 **response**: HTTP/1.1 206 Partial Content (text/texmacs)
GET5	2,802	3,156	**GET** http://rfqdn3.cdn.ab.sk/motoTrip/fileSequence1.ts HTTP/1.1 **response**: HTTP/1.1 302 Moved Temporarily
GET6	23872,649	16195,253	**GET** http://se3.se.rfqdn3.cdn.ab.sk/motoTrip/fileSequence1.ts HTTP/1.1 **response**: HTTP/1.1 206 Partial Content (text/texmacs)

Figure 44: *Table of requests to Cisco CDS in Bratislava.*

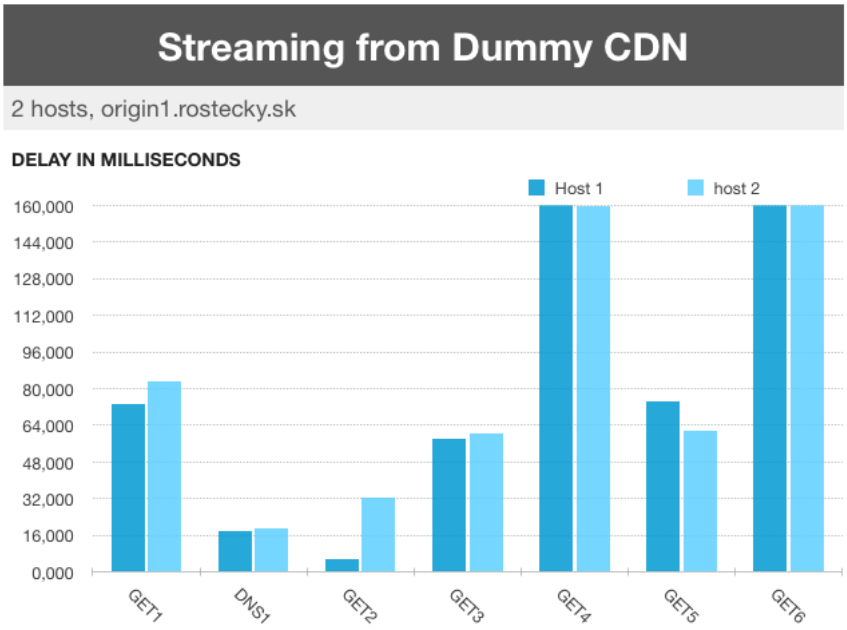

Figure 45: *Graph of requests needfed to reached content on Dummy CDN in Bratislava.*

	HOST 1	HOST 2	REQUESTS
GET1	73,165	83,221	GET http://147.175.15.42/CDNi2/dummyCdn/rfqdn1/motoTrip/medium/mototrip2013.m3u8 HTTP/1.1 **response**: HTTP/1.1 302 Moved Temporarily
DNS1	18,082	19,356	Standard query A origin1.rostecky.sk
GET2	5,428	32,616	**GET** http://origin1.rostecky.sk/motoTrip/medium/mototrip2013.m3u8 HTTP/1.1 **response**: HTTP/1.1 206 Partial Content (text/plain)
GET3	58,417	60,389	**GET** http://147.175.15.42/CDNi2/dummyCdn/rfqdn1/motoTrip/medium/fileSequence0.ts HTTP/1.1 **response**: HTTP/1.1 302 Moved Temporarily
GET4	470,712	9324,509	**GET** http://origin1.rostecky.sk/motoTrip/medium/fileSequence0.ts HTTP/1.1 **response**: HTTP/1.1 206 Partial Content (text/texmacs)
GET5	74,487	61,690	**GET** http://147.175.15.42/CDNi2/dummyCdn/rfqdn1/motoTrip/medium/fileSequence1.ts HTTP/1.1 **response**: HTTP/1.1 302 Moved Temporarily
GET6	357,854	1223,398	**GET** http://origin1.rostecky.sk/motoTrip/medium/fileSequence1.ts HTTP/1.1 **response**: HTTP/1.1 206 Partial Content (text/texmacs)

Figure 46: *Table of requests to Dummy CDN in Bratislava.*

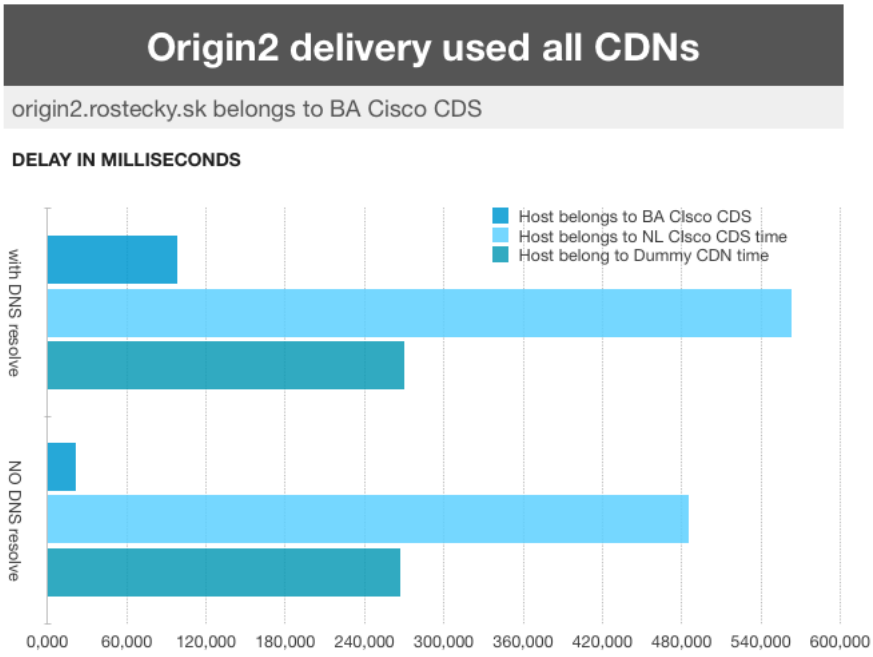

Figure 47: *Graph of average delay to reached content.*

HOST BELONGS TO BA CISCO CDS	
TIME	**REQUESTS**
28,603	Standard query 0x6675 A rfqdn3.cdn.ab.sk
	Standard query response 0x6675 A 147.175.204.110
2,591	GET http://rfqdn3.cdn.ab.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
48,498	Standard query 0x9ef7 A se3.se.rfqdn3.cdn.ab.sk
	Standard query response 0x9ef7 A 147.175.204.122
16,711	GET http://se3.se.rfqdn3.cdn.ab.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 206 Partial Content (text/plain)
1,870	GET http://rfqdn3.cdn.ab.sk/motoTrip/fileSequence0.ts HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
0	GET http://se3.se.rfqdn3.cdn.ab.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1
98,273	Time to first request directly to content
21,172	with out DNS resolve

Figure 48: *Table of requests with average delays from host belongs to BA Cisco CDS.*

HOST BELONGS TO NL CISCO CDS	
TIME	**REQUESTS**
33,841	Standard query 0xa1b3 A rfqdn3.cdn.ab.sk
	Standard query response 0xa1b3 A 147.175.204.110
206,921	GET http://rfqdn3.cdn.ab.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
26,790	Standard query 0xa5f3 A rfqdn6.cdn.rostecky.sk
	Standard query response 0xa5f3 A 139.63.240.12
22,277	GET http://rfqdn6.cdn.rostecky.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
17,576	Standard query 0xd057 A se1.se.rfqdn6.cdn.rostecky.sk
	Standard query response 0xd057 A 139.63.240.11
49,556	GET http://se1.se.rfqdn6.cdn.rostecky.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 206 Partial Content (text/plain)
182,865	GET http://rfqdn3.cdn.ab.sk/motoTrip/fileSequence0.ts HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
23,352	GET http://rfqdn6.cdn.rostecky.sk/motoTrip/fileSequence0.ts HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
0	GET http://se1.se.rfqdn6.cdn.rostecky.sk/motoTrip/fileSequence0.ts HTTP/1.1
563,177	Time to first request directly to content
484,970	with out DNS resolve

Figure 49: *Table of requests with average delays from host belongs to NL Cisco CDS.*

HOST BELONG TO DUMMY CDN	
TIME	**REQUESTS**
1,412	Standard query 0xbb13 A rfqdn3.cdn.ab.sk
	Standard query response 0xbb13 A 147.175.204.110
74,927	GET http://rfqdn3.cdn.ab.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
71,080	GET http://147.175.15.42/CDNi2/dummyCdn/rfqdn3/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
1,940	Standard query 0x5428 A origin2.rostecky.sk
	Standard query response 0x5428 A 147.175.15.43
5,154	GET http://origin2.rostecky.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 206 Partial Content (text/plain)
49,209	GET http://rfqdn3.cdn.ab.sk/motoTrip/fileSequence0.ts HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
66,164	GET http://147.175.15.42/CDNi2/dummyCdn/rfqdn3/motoTrip/fileSequence0.ts HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
0	GET http://origin2.rostecky.sk/motoTrip/fileSequence0.ts HTTP/1.1
269,886	Time to first request directly to content
266,534	with out DNS resolve

Figure 50: *Table of requests with average delays from host belongs to Dummy CDN.*

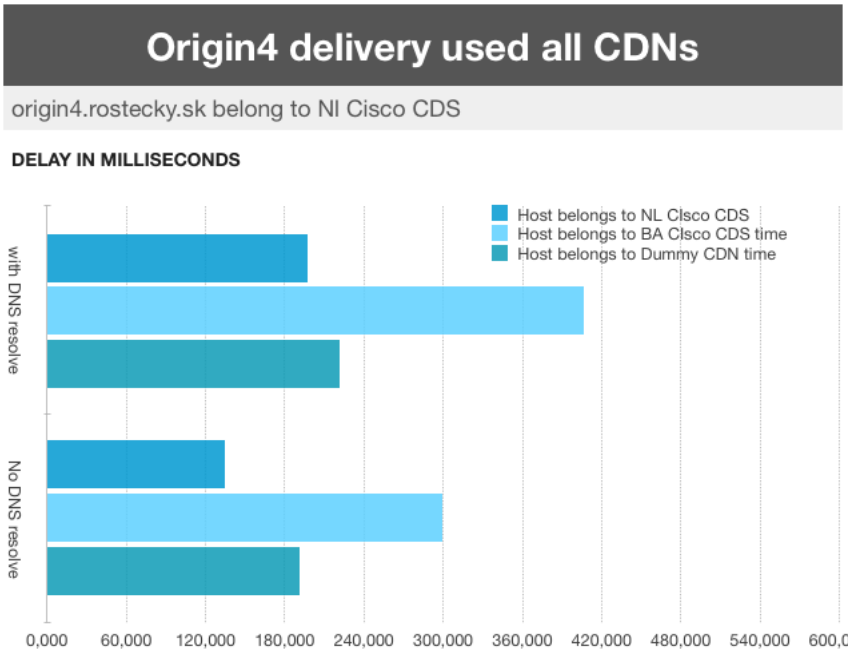

Figure 51: *Graph of avarage delay to reached content.*

HOST BELONGS TO NL CISCO CDS	
TIME	**REQUESTS**
26,606	Standard query 0x5f26 A rfqdn4.cdn.rostecky.sk
	Standard query response 0x5f26 A 139.63.240.12
21,962	GET http://rfqdn4.cdn.rostecky.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
35,630	Standard query 0xaf4d A se1.se.rfqdn4.cdn.rostecky.sk
	Standard query response 0xaf4d A 139.63.240.11
90,396	GET http://se1.se.rfqdn4.cdn.rostecky.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 206 Partial Content (text/plain)
22,187	GET http://rfqdn4.cdn.rostecky.sk/motoTrip/fileSequence0.ts HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
0	GET http://se1.se.rfqdn4.cdn.rostecky.sk/motoTrip/fileSequence0.ts HTTP/1.1
196,781	Time to first request directly to content
134,545	with out DNS resolve

Figure 52: *Table of requests with average delays from host belongs to NL Cisco CDS.*

HOST BELONGS TO BA CISCO CDS	
TIME	**REQUESTS**
17,654	Standard query 0x5b22 A rfqdn4.cdn.rostecky.sk
	Standard query response 0x5b22 A 139.63.240.12
152,365	GET http://rfqdn4.cdn.rostecky.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
35,477	Standard query 0x96b9 A rfqdn6.cdn.ab.sk
	Standard query response 0x96b9 A 147.175.204.110
2,588	GET http://rfqdn6.cdn.ab.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
53,660	Standard query 0xc807 A se3.se.rfqdn6.cdn.ab.sk
	Standard query response 0xc807 A 147.175.204.122
11,389	GET http://se3.se.rfqdn6.cdn.ab.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 206 Partial Content (text/plain)
128,306	GET http://rfqdn4.cdn.rostecky.sk/motoTrip/fileSequence0.ts HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
4,922	GET http://rfqdn6.cdn.ab.sk/motoTrip/fileSequence0.ts HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
0	GET http://se3.se.rfqdn6.cdn.ab.sk/motoTrip/fileSequence0.ts HTTP/1.1
406,361	Time to first request directly to content
299,570	with out DNS resolve

Figure 53: *Table of requests with average delays from host belongs to BA Cisco CDS.*

HOST BELONGS TO DUMMY CDN	
TIME	**REQUESTS**
27,533	Standard query 0x8323 A rfqdn4.cdn.rostecky.sk
	Standard query response 0x8323 A 139.63.240.12
0,000	GET http://rfqdn3.cdn.ab.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
71,080	GET http://147.175.15.42/CDNi2/dummyCdn/rfqdn3/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
1,940	Standard query 0x213d A origin4.rostecky.sk
	Standard query response 0x213d A 147.175.15.94
5,154	GET http://origin2.rostecky.sk/motoTrip/mototrip2013.m3u8 HTTP/1.1
	HTTP/1.1 206 Partial Content (text/plain)
49,209	GET http://rfqdn3.cdn.ab.sk/motoTrip/fileSequence0.ts HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
66,164	GET http://147.175.15.42/CDNi2/dummyCdn/rfqdn3/motoTrip/fileSequence0.ts HTTP/1.1
	HTTP/1.1 302 Moved Temporarily
0	GET http://origin2.rostecky.sk/motoTrip/fileSequence0.ts HTTP/1.1
221,080	Time to first request directly to content
191,607	with out DNS resolve

Figure 54: *Table of requests with average delays from host belongs to Dummy CDN.*

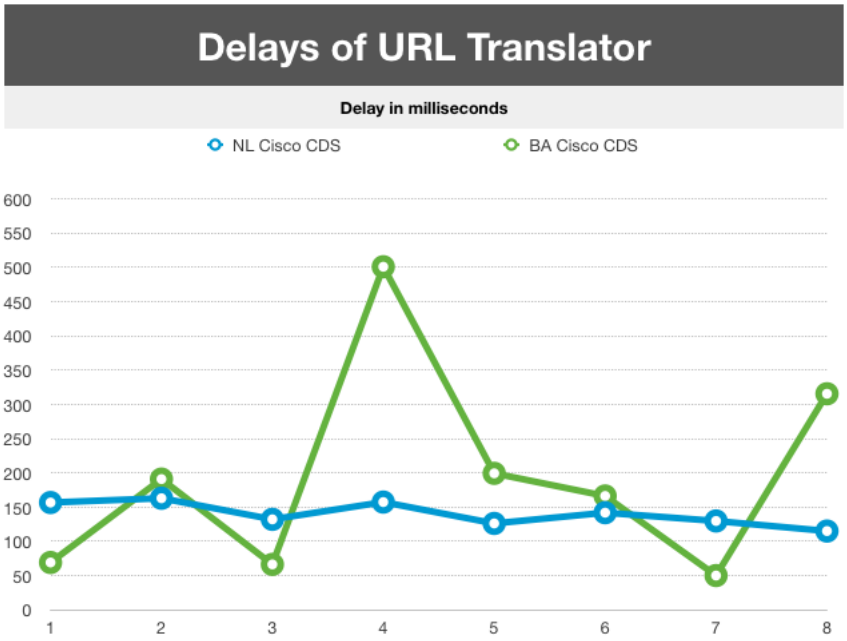

Figure 55: *Delays of requests that contain URL translations.*

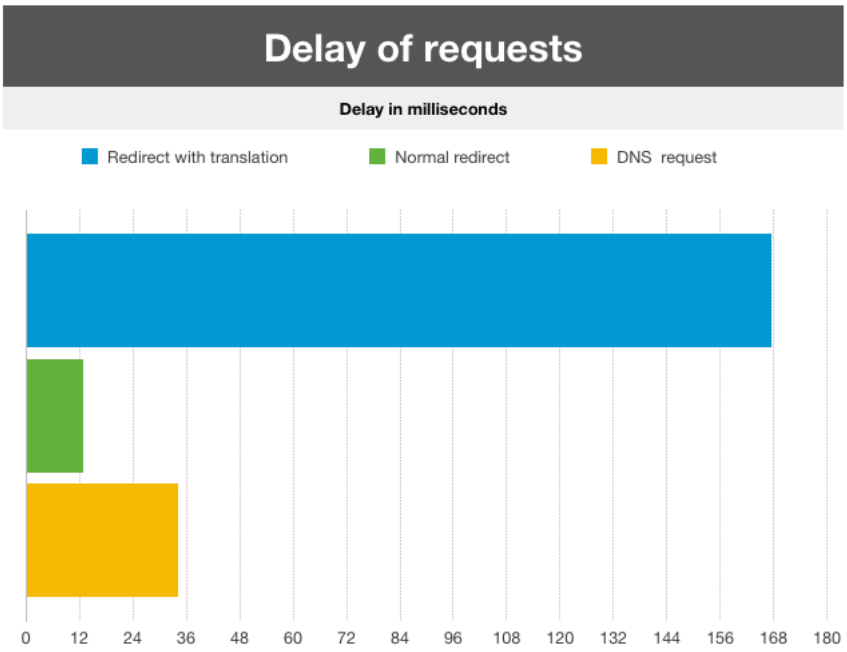

Figure 56: *Compare average delays of requests.*

6 Conslusion

Our master thesis is focused on creating service for users that will offer user to mediate multimedia content with guaranteed quality of delivering content. Guaranteed quality of delivering should be provided by content delivery network. In later parts of our work, we focused deeper on content delivery networks and guaranteed quality delivering of content. We created adapter to interconnect two CDNs that increase performance and quality of delivering content. Adapter consists of two interfaces that are implemented as web application with user interface to manage it. Administrator of content delivery network can easily set up CDN interface and create interconnection with other CDNs without changes on origins servers such as URLs. Our interface is necessary only attach to existing infrastructure of content delivery network. We verified our implementation and created set of tests to found delays in communication with our CDN adapter. We rate results of tests positive. We compare delays to first request that returns content between delivering content used one CDN and used interconnected CDNs. Every Delay was increase by time needful to translation of URL and one more redirection. Time for translation URL was average about 167ms and one average redirection is 13ms. Average time what increase every request to content is 180ms.

We see perspective to continue this project in future. For now, project is functional prototype of interconnection and demonstrate possibility of interconnect CDNs. There are a more ways how improve this CDN interface. For example we can reduce time needful to translation used by PHP accelerators. Next we can implement the other content delivery networks such as Jet-Stream. We can implement prefetched caching content after create interconnection and more.

References

[1] Samsung Electronics Co., Ltd.: *Samsung Smart TV Developer Documentation 3.5.*; 2012. [Online] Cited: 2013-05-03. http://www.samsungdforum.com/upload-files/files/guide/data/html/html-3/cover-page.html.

[2] Samsung Electronics Co., Ltd.: *Development Guide*; 2013. [Online] Cited: 2013-05-03 http://www.samsungdforum.com/Guide/rel00010/index.html

[3] Microsoft: *Smooth Streaming*; 2013. [Online] Cited: 2013-05-05 http://www.iis.net/downloads/microsoft/smooth-streaming

[4] Thorsten Lohmar Torbjorn Einarsson Per Frojdh Frederic Gabin Markus Kampmann: *Adaptive HTTP Streaming of Live Content*; 2011. IEEE, 2011, 978-1-4577-0351-5

[5] Moth, David: *Majority of TV viewing now on-demand for one in four Brits*; 2012-04-30. [Online] Cited: 2013-05-04 http://econsultancy.com/sk/blog/9726-majority-of-tv-viewing-now-on-demand- for-one-in-four-brits

[6] Delboy: *Smart Internet TV Statistics Reveal Poor App Take-Up.*; 2012-12-28. [Online] Cited: 2013-04-08 http://www.worldtvpc.com/blog/smart-internet-tv-statistics-reveal-poor-app-take-up

[7] Technologies, Akamai: *Media Streaming Glossary*; 2013. [Online] Cited: 2013-04-30 http://www.akamai.com/html/solutions/sola-cdn.html

[8] Athena Vakali and George Pallis: *Content Delivery Networks: Status and Trends*; 2003. IEEE INTERNET COMPUTING, 2003, Aristotle University of Thessaloniki

[9] Rajkumar Buyya, Mukaddim Pathan, Athena Vakali: *Content Delivery Networks*; 2008. 2008, Springer, ISBN 978-3-540-77886-8

[10] Paul Barford, Jin-Yi Cai, Jim Gast: *Cache Placement Methods Based on Client Demand*; 2002. IEEE INFOCOM, 2002

[11] Gilmore, Patrick W.: *Peering with Content Distribution Networks*; 2008-21-05. UKNOF. http://www.uknof.org.uk/uknof10/Gilmore-CDN-peering.pdf

[12] Harry McCracken: *Who's Winning, iOS or Android? All the Numbers, All in One Place.*; 2013-04-16. [Online] Cited: 2013-04-30 http://techland.time.com/2013/04/16/ios-vs-android/

[13] Khalaf, Simon: *Flurry Five-Year Report: It's an App World. The Web Just Lives in It*; 2013-03-04. [Online] Cited: 2013-04-30 http://blog.flurry.com/bid/95723/Flurry-Five-Year-Report-It-s-an-App- World-The-Web-Just-Lives-in-It

[14] Google inc: *Dashboards*; 2013-01-05. [Online] Cited: 2013-05-06 http://developer.android.com/about/dashboards/index.html

[15] Google inc: *Ice Cream Sandwitch*; 2012. [Online] Cited: 2013-05-06 http://developer.android.com/about/versions/android-4.0- highlights.html

[16] ART LEE: *Android Goes Beyond Google*; 2012-05. VIOSOFT. [Online] Cited: 2013-05-05 http://rtcmagazine.com/articles/view/102586

[17] Khronos Group: *OpenMAX AL - The Standard for Media Library Portability*; 2013. [Online] Cited: 2013-05-05 http://www.khronos.org/openmax/al/

[18] Google inc: *Android Supported Media Formats*; 2012. [Online] Cited: 2013-05-05 http://developer.android.com/guide/appendix/media-formats.html

[19] Naina Khedekar: *Google Play Movies Review*; 2013-04-02. [Online] Cited: 2013-05-05 http://tech2.in.com/reviews/apps/google-play-movies-review/860392

[20] Amazon.com: *LOVEFiLM*; 2013. [Online] Cited: 2013-05-05 http://www.lovefilm.com/

[21] European Telecommunications Standards Institute: *CDN Interconnection Architecture*; 2013-04. ETSI TS 182 032, V1.1.1, 2013 © European Telecommunications Standards Institute 2013

[22] European Telecommunications Standards Institute: *Media Content Distribution (MCD): CDN Interconnection, use cases and requirements*; 2012-11. ETSI TS 102 990, V1.1.1, 2012 © European Telecommunications Standards Institute 2012

[23] Wowza Media Systems, LLC: *Wowza media system*; 2013 [Online] Cited: 2013-11-20 http://www.wowza.com/

[24] VideoLan organization: *VLC media player*; 2013. [Online] Cited: 2013-11-27 http://www.videolan.org/vlc/

[25] Cisco Systems, Inc.: *Cisco Internet Streamer CDS 3.1 Software Configuration Guide*; 2013-01. OL-27552-02, 2013

[26] Yii Software LLC: *The Fast, Secure and Professional PHP Framework*; 2013. [Online] Cited: 2013-12-03 http://www.yiiframework.com/

[27] Peter Payter Gašparík: *Vel'ký prehl'ad najpoužívanejších PHP frameworkov*; 2012-09-05. [Online] Cited: 2013-11-28 http://www.zajtra.sk/programovanie/820/velky-prehlad-najpouzivanejsich-php-frameworkov

[28] Herve: *gwebservice*; 2013. [Online] Cited: 2013-12-02 http://www.yiiframework.com/extension/gwebservice/

[29] W3C: *SOAP Version 1.2*; 2004. [Online] Cited: 2013-11-23 http://www.w3.org/TR/soap/

[30] Cisco Systems, Inc: *Cisco Videoscape Distribution Suite Service Broker Data Sheet*; 2013. [Online] Cited: 2014-5-5 http://www.cisco.com/c/en/us/products/collateral/video/videoscape-distribution-suite-service-broker/data_sheet_c78-713721.html

[31] Cisco Systems, Inc: *Cisco VDS Service Broker 1.0.1 Software Configuration Guide*; 2013. [Online] Cited: 2014-5-5 http://www.cisco.com/c/en/us/td/docs/video/videoscape/distribution_suite/SB/1-0-0/ConfigurationGuide/vds_sb_1_0_1_config.html

[32] Cisco Systems, Inc: *Cisco Internet Streamer CDS 3.2 API Guide*; 2013. [Online] Cited: 2014-5-5 http://www.cisco.com/c/en/us/td/docs/video/cds/cda/is/3_2/developer_guide/iscds32APIGuide.html

[33] Cisco Systems, Inc: *Cisco Internet Streamer CDS 3.2 Software Configuration Guide*; 2013. [Online] Cited: 2014-5-5 http://www.cisco.com/c/en/us/td/docs/video/cds/cda/is/3_2/configuration_guide/ISCDS3-2config.html

A Technical documentation

Implementation of CDN interface is done as web-based application. This application is implemented in PHP Framework YII and this application used MySQL database. On server run Debian GNU/Linux and for web server we used Apache 2. Structure of application directories start int root directory of web server "/var/www". There are several directories on test server but one CDNi2 directory and .htaccess are the most important. CDNi2 directory contain YII application and .htaccess provides redirection of request from Cisco CDS to translate URL to our Yii application. Yii application contain a few directories but all what we need we can found in /protected/config, /protected/controllers, /protected/models and /protected/views. There are all files with classes and functions that create a logic of our application. The most interested classes in application are:

- class EndpointRemoteController
- class SoapController
- class UrlTranslationServiceController
- class CiscoCDS

Class **EndpointRemoteController** contain probably the all main logic of CDNi interface. From this class are initiate function to create, update and delete CDNi interconnection. In Example 6 is shown function `actionCreate()` from EndpointRemoteController class. This function creates offer from downstream CDN to upstream CDN to create interconnection between them.

Example 6: *Function actionCreate() from EndpointRemoteContoller*

```
public function actionCreate() {
    $model = new EndpointRemote;
    $modelLocal = EndpointLocal::getLocalEndpoint();
    if (isset($_POST['EndpointRemote'])) {
        if($_POST['EndpointRemote']['url'] == $modelLocal->url){
            echo "CAN NOT CREATE INTERCONNECTION WITH SAME URL AS IS LOCAL URL";
            exit;
        }
        $model->setAttributes($_POST['EndpointRemote']);
        $model->endpoint_status_id = 2;
        $model->endpoint_type_id = 2;
        if ($model->saveWithRelated($relatedData)) {
            $id = $model->getPrimaryKey();
            $client = $this->createSoapClient($id);
            $client->setOffer($modelLocal->name, $modelLocal->url, $modelLocal->priority,↩
              » 2, 1, $modelLocal->endpoint_gateway_type_id, $modelLocal->url_translator↩
              » );
            $client->logging('Receive offer from ' . $modelLocal->name . ':' . ↩
              » $modelLocal->url);
            Log::logging("Send offer to " . $model->name . ':' . $model->url);
            $arrayFootprints = array();
            $arrayFootprints = Footprint::model()->findAllByAttributes( array('↩
              » endpoint_id' => $modelLocal->id ));
            foreach ($arrayFootprints as $key => $value) {
```

```
                $client->setFootprints($modelLocal->name, $modelLocal->url, $value->↩
                   » subnet_num, $value->mask_num, $value->subnet_ip, $value->prefix);
                $client->logging('Receive footprint ' . $value->subnet_ip . '/' . $value↩
                   » ->prefix . ' from ' . $modelLocal->name . ':' . $modelLocal->url);
                Log::logging('Send footprint ' . $value->subnet_ip . '/' . $value->prefix↩
                   »  . ' to ' . $model->name . ':' . $model->url);
            }
            $modelCapabilities = EndpointCapability::model()->findAllByAttributes(array('↩
               » endpoint_id' => $modelLocal->id));
            $capability = array();
            foreach ($modelCapabilities as $key => $value) {
                $capability[] = $value->capability_id;
            }
            $client->setCapabilities( $modelLocal->name, $modelLocal->url, $capability);
            unset($client);
            if (Yii::app()->getRequest()->getIsAjaxRequest())
                Yii::app()->end();
            else
                $this->redirect(array('view', 'id' => $model->id));
        }
    }
    $this->render('create', array( 'model' => $model));
}
```

In Example 7 is shown part of function `actionAcceptOffer()` from EndpointRemoteController class. This function accept receive offer to create CDNi interconnection from downstream CDN. Shown Part of function switch statement that contain creating Content Origin and other necessary procedures to set up CDN to delivery content.

Example 7: *Part of function actionAcceptOffer() from EndpointRemoteContoller*

```
    switch ($model->endpoint_gateway_type_id) {
        case 1: // DUMMY
            foreach ($contentOrigins as $keyContentOrigin => $valueContentOrigin) {
                $listFqdn = array();
                $listFqdn = $client->getListFqdn();
                if(empty($listFqdn)){
                    $client->logging('Can not create Content Origin, all fqdn are used.')↩
                       » ;
                    Log::logging('Remote CDNi haven\' got any free fqdn.');
                }
                Log::logging('DEBUG: listFqdn: ' . print_r($listFqdn, TRUE));
                foreach ($listFqdn as $keyFqdn => $valueFqdn) {
                    if($client->createDummyContentOrigin($modelLocal->name, $modelLocal->↩
                       » url, $valueContentOrigin['name'], $valueContentOrigin['↩
                       » originServer'], $valueFqdn['fqdn'] )){
                        $client->logging("Create Content ORIGIN initialized by: " . ↩
                           » $modelLocal->url );
                        Log::logging("Create content ORIGIN in " . $model->url );
                    } else {
```

```
                    $client->logging("ERROR: unsuccesfull Create Content ORIGIN ↩
                       » initialized by: " . $modelLocal->url );
                    Log::logging("ERROR: unsuccesfull Create content ORIGIN in " . ↩
                       » $model->url );
                }
            break;
            }
        }
    break;
    case 2: // CISCO CDS
        foreach ($contentOrigins as $key => $value) {
        $listFqdn = array();
        $listFqdn = $client->getListFqdn();
        if(empty($listFqdn)){
            $client->logging('Can not create Content Origin, all fqdn are used.');
            Log::logging('Remote CDNi haven\' got any free fqdn.');
        }
        foreach ($listFqdn as $keyFqdn => $valueFqdn) {
            $idContentOrigin = $client->createContentOrigin($modelLocal->name, ↩
               » $modelLocal->url, $value['name'], $value['originServer'], $valueFqdn↩
               » ['fqdn'] );
            if(!empty($idContentOrigin)){
                $client->logging("Create Content ORIGIN initialized by: " . ↩
                   » $modelLocal->url . ":" . $idContentOrigin);
                            Log::logging("Create content ORIGIN in " . $model->↩
                               » url . ":" . $idContentOrigin);
                $idDeliveryService = $client->createDeliveryService($modelLocal->name↩
                   » , $modelLocal->url, $value['name'], $idContentOrigin);
                if(!empty($idDeliveryService)){
                    $client->logging("Create Delivery Service initialized by: "  . ↩
                       » $modelLocal->url . ":" . $idDeliveryService);
                    Log::logging("Create Delivery Service in " . $model->url . ":" . ↩
                       » $idDeliveryService);
                    $devices = array();
                    $devices = $client->getDevices('SE');
                    if(empty($devices)){
                    Log::logging("ERROR: getDevices is Empty");
                    EndpointRemote::setErrorStatus($model->id);
                    $client->setEndpointRemoteErrorStatus($modelLocal->name, ↩
                       » $modelLocal->url);
                    } else {
                        Log::logging("DEBUG: getDevice return: " . print_r($devices, ↩
                           » TRUE));
                        foreach ($devices as $keyDevice => $valueDevice) {
                            if($valueDevice['status'] == 'Online'){
                                Log::logging("DEBUG: assign: " . $valueDevice['id']);
                                    if($client->assignSEs($modelLocal->name, ↩
                                       » $modelLocal->url, $idDeliveryService, ↩
                                       » $valueDevice['id'])){
                                        Log::logging("Assign Service Engine in " . ↩
                                           » $model->url . ":" . $idDeliveryService);
                                        $client->logging("Assign Service Engine ↩
                                           » initialized by: "  . $modelLocal->url . "↩
```

```
                                        » :" . $idDeliveryService);
                                    } else {
                                        ... error logging
                                    }
                                    break;
                                }
                            }
                        } else {
                            ... error logging
                        }
                    } else {
                        ... error logging
                    }
                    break;
                }
            }
        break;
        default:
            ... error logging
            break;
    }
```

Class **SoapController** provide web service of our application. In Example 8 is shown how is necessary define function and what is generated from it.

Example 8: *Definition function for web service and notion of wsdl schema*

```
// DEFINITION FUNCTION  FOR WEB SERVICE
    /**
     * Assign Service Engine to Delivery Service in CISCO CDS
     * @param string    $name
     * @param string    $url
     * @param string    $deliveryServiceID
     * @param string    $contentAcquirer
     * @return bool     $status
     * @soap
     */
    public function assignSEs($name, $url, $deliveryServiceID, $contentAcquirer){
        $cisco = new CiscoCDS;
        $status = FALSE;
        $model = EndpointRemote::model()->findByAttributes(array('name' => $name, 'url' ↩
           » => $url));
        $status = $cisco->assignSEs($deliveryServiceID, $contentAcquirer);
        if($status){
            $params = array('contentAcquirer' => $contentAcquirer, );
            CdniAction::addAction($model->id, 'assignSEs', $params);
            return TRUE;
        }
        return FALSE;
    }
```

```
// GENERATED WSDL FILE FOR WEB SERVICE
<definitions xmlns="http://schemas.xmlsoap.org/wsdl/" xmlns:tns="urn:SoapControllerwsdl" ↩
   » xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns:xsd="http://www.w3.org/2001/↩
   » XMLSchema" xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/" xmlns:soap-enc="http://↩
   » schemas.xmlsoap.org/soap/encoding/" name="SoapController" targetNamespace="urn:↩
   » SoapControllerwsdl">
<script id="tinyhippos-injected"/>
...
    <wsdl:message name="assignSEsRequest">
        <wsdl:part name="name" type="xsd:string"/>
        <wsdl:part name="url" type="xsd:string"/>
        <wsdl:part name="deliveryServiceID" type="xsd:string"/>
        <wsdl:part name="contentAcquirer" type="xsd:string"/>
    </wsdl:message>
    <wsdl:message name="assignSEsResponse">
        <wsdl:part name="return" type="xsd:boolean"/>
    </wsdl:message>
...
    <wsdl:operation name="assignSEs">
        <wsdl:documentation>
            Assign Service Engine to Delivery Service in CISCO CDS
        </wsdl:documentation>
        <wsdl:input message="tns:assignSEsRequest"/>
        <wsdl:output message="tns:assignSEsResponse"/>
    </wsdl:operation>
...
    <wsdl:operation name="assignSEs">
        <soap:operation soapAction="urn:SoapControllerwsdl#assignSEs" style="rpc"/>
        <wsdl:input>
            <soap:body use="encoded" namespace="urn:SoapControllerwsdl" encodingStyle="↩
               » http://schemas.xmlsoap.org/soap/encoding/"/>
        </wsdl:input>
        <wsdl:output>
            <soap:body use="encoded" namespace="urn:SoapControllerwsdl" encodingStyle="↩
               » http://schemas.xmlsoap.org/soap/encoding/"/>
        </wsdl:output>
    </wsdl:operation>
...
</definitions>
```

Class **UrlTranslationServiceController** provide translation service for Cisco CDS. In Example 9 is shown part of function that find appropriate CDN and translate URL.

Example 9: *Part of function to find appropriate CDN and translate URL*

```
$params = unserialize($valueModelAction->params);
foreach ($params as $keyPar => $valuePar) {
    if ( $valuePar['fqdn'] == $fqdn ){
        $originServer = $valuePar['originServer'];
        $originName = $valuePar['name'];
```

```
    }
}
$modelsEndRe = EndpointRemote::model()->findAllByAttributes(array('endpoint_status_id' =>↩
   »  '6'));
foreach ($modelsEndRe as $keyEndRe => $modelEndRe) {
    $arrayFootprints = Footprint::model()->findAllByAttributes( array('endpoint_id' => ↩
       » $modelEndRe->id ));
    foreach ($arrayFootprints as $keyFootprit => $footprint) {
        if($requestIpLong >= $footprint->subnet_num && $requestIpLong < ($footprint->↩
           » subnet_num + pow(2, 32 - $footprint->prefix) )){
            $endpointRemoteArray[] = array(
                'idEndpointRemote' => $modelEndRe->id,
                'subnet_num' => $footprint->subnet_num,
                'prefix' => $footprint->prefix,
                'fqdn' => '',
            );
        }
    }
}
if( count($endpointRemoteArray) > 0){
    foreach ( $endpointRemoteArray as $key => $value ) {
        $modelCdniAction = CdniAction::model()->findByAttributes(array('endpoint_id' => ↩
           » $value['idEndpointRemote'] , 'action' => 'RemoteAssociateOriginToFqdn'));
        $unserializepParams = array();
        $unserializepParams = unserialize($modelCdniAction->params);
        foreach ($unserializepParams as $keyParams => $valueParams) {
            if ($originName == $valueParams['name'] && $originServer == $valueParams['↩
               » originServer'] ) {
                $endpointRemoteArray[$key]['fqdn'] = $valueParams['fqdn'];
            }
        }
    }
    $hihterPrefix = 0;
    foreach ($endpointRemoteArray as $key => $value) {
        if( $value['fqdn'] != '' ){
            if( $value['prefix'] > $hihterPrefix ){
                $hihterPrefix = $value['prefix'];
                $fqdn = $value['fqdn'];
            }
        }
    }
} else {
    // we havent got any match we have to handle requst, therefore we redirect requste on↩
       »  some SE of our CDN
}
if($fqdn == '') {
    Log::logging('ERROR: UrlTranslationService is not setUp fqdn!');
    die();
} else {
    Header('Content-type: text/xml');
    $url = $protocol . $fqdn . $content;
    $url = htmlspecialchars($url);
    echo '
```

```
        <?xml version="1.0" encoding="UTF-8"?>
        <SOAP-ENV:Envelope  xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
                xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
                xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                xmlns:xsd="http://www.w3.org/2001/XMLSchema"
                xmlns:CDNUTNS1="http://cisco/CDS/CDNUrlTranslation"
                xmlns:CDNUTNS2="http://schemas.cisco/CDS/CDNUrlTranslation/Schema">
        <SOAP-ENV:Body>
            <CDNUTNS2:UrlTranslationResponse>
                <TranslatedUrl>'.$url.'</TranslatedUrl>
                <SignUrl>false</SignUrl>
                </CDNUTNS2:UrlTranslationResponse>
            </SOAP-ENV:Body>
        </SOAP-ENV:Envelope>';
    die();
}
```

Class **CiscoCDS** contain implemented Cisco CDS API. In Example 10 is shown function to assign Service Engines to delivery service.

Example 10: *Functions to assign Service Engines to delivery service*

```
private function makeRequest($taskAPI,$action,$params) {
    $modelLocal = EndpointLocal::getLocalEndpoint();
    $params['action']=$action;
    $urlString =  $modelLocal->url_cdn . ":" . $modelLocal->port_cdn . "/servlet/".↩
       » $taskAPI."?"  .  http_build_query($params);// + "&param=" + channelId_;
    $response = http_get(
    $urlString,
    array (
        'httpauth' => $modelLocal->login . ':' . $modelLocal->pass,
        'httpauthtype' => HTTP_AUTH_BASIC
    ));
    $responseObj=http_parse_message($response);// var_dump($responseObj);
    if ($responseObj -> responseCode != 200 ) return false;
    $body=$responseObj->body;
    file_put_contents(__DIR__."/resp/".$action.".resp.xml",$body);
    $xml=new SimpleXMLElement($body);
    return $xml;
}
public function assignSEs($deliveryServiceID, $contentAcquirer) {
    $params = array();
    $params ['deliveryService'] = $deliveryServiceID;
    $params ['contentAcquirer'] = $contentAcquirer;
    $params ['se'] = 'all';
    $resXML = $this->makeRequest('com.cisco.unicorn.ui.ChannelApiServlet', 'assignSEs', ↩
       » $params);
    if($resXML->message["status"] == "success"){
        return TRUE;
    } else {
```

```
        Log::logging("ERROR: assignSEs: " . $resXML->message["status"] . ":" . $resXML->↩
            » message["message"]);
        return FALSE;
    }
    return FALSE;
}
```

B User Guide

Our implementation of CDN interface is a web based application. During our testing and developing we have three instances of CDN interface that are gateways to three content delivery networks. We demonstrate configuration of one of them. After typing URL "http://147.175.15.41/CDNi2/" to web browser we can see home page of CDN interface like is shown on Figure 57.

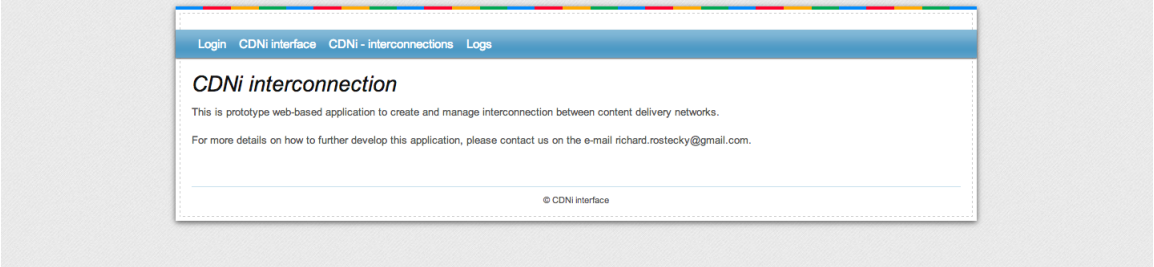

Figure 57: *Home page of CDN interface.*

This page has an information character. For all the other operation in CDN interface we have to be logged in to the application. CDN interface contains only one user. Default user login is "admin" and password is "admin". Login screen is shown on every menu item until we are not logged in to application. On Figure 58 is shown login page.

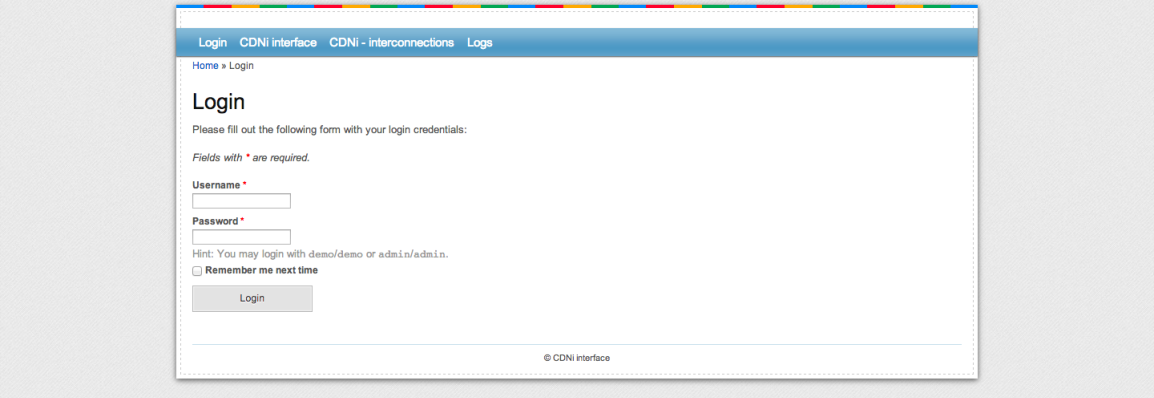

Figure 58: *Login page of CDN interface.*

After successful log in to application we have to set up CDN interface. We can click on CDN interface menu item and we can see page shown on Figure 59.

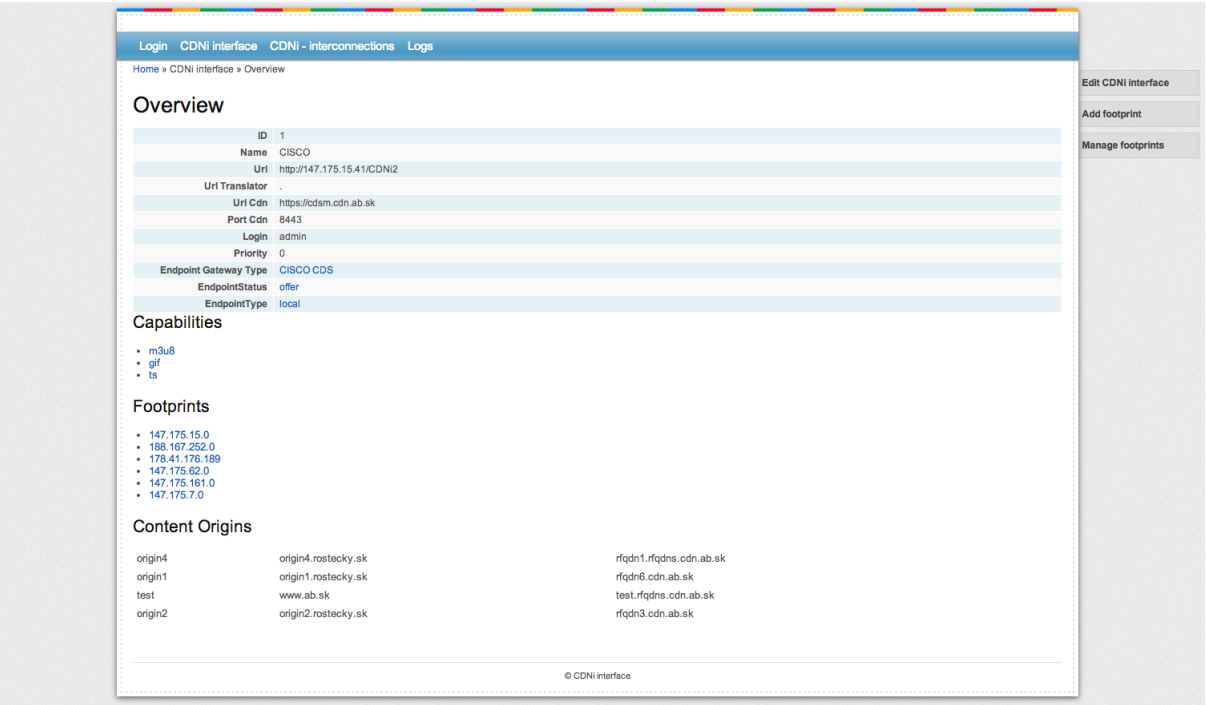

Figure 59: *Overview of settings of CDN interface.*

This page contains overview of settings of CDN interface. On the top we can see table with base information. Under table we can see capabilities of content delivery network for that CDN interface is gateway. Under capabilities are footprints that content delivery network can handle. As last item that we can see on the page is a list of Content Origins. Some Content Origins are learned from the other content delivery networks and some are defined by content delivery network itself. Base settings and capabilities are possible to set up after click on "Edit CDNi interface". There are some required fields. Interconnection is unique identified by Name and URL. Next field is URL Translator. URL translator filed is prepare for other implementation of content delivery networks. It is necessary to set up URL translator for Cisco CDS by using CDS manager. Next fields are URL cdn and Port cdn. These field are necessary to set up if CDN interface is gateway for Cisco CDS. Cisco API is reachable on these url and port. The last two fields are Login and Password. These parameters are used for authenticatation to Cisco CDS for accessing to use Cisco API. The last settings on actual page are check-boxes that describe capabilities of content delivery network. We can add footprints after click on "Add footprint" item in right menu. After click we can see easy form that contains two fields. One of them represents ip subnet and second field represents prefix of the subnet. After click on "save" button footprint will be added. We can manage footprints after click on "Manage Footprints" item in right menu. On Figure 60 is shown page to manage footprints. On this page are shown all footprints that contain CDN interface include footprints learned from interconnected content delivery networks. Administrator can manage footprints that belong to other CDN. It is benefit if administrator want to use only some footprints from remote CDN, no all of them. To set up Content origins is necessary to use default manager that content delivery network contains. In our case with Cisco CDS it is CDS manager. There we can set up Content origins. In case with Dummy CDN we have to edit source code,

because purpose of Dummy CDN is only for laboratory tests.

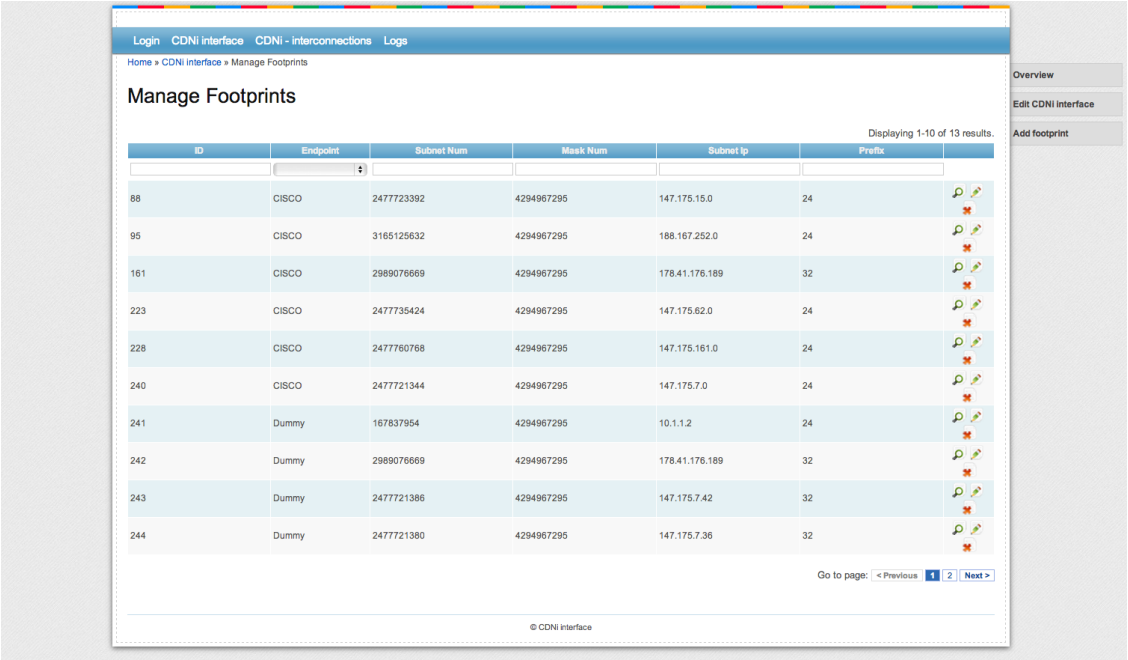

Figure 60: *Page of managing footprints.*

After seting up CDN interface we can create CDNi interconnection. We can click on "CDNi interconnection" item in Top menu. On Figure 61 is shown page that we can see after the click. On the top of page is filter to filtering the table under them. The table contains interconnections. In table we can see Name and URL of remote CDN interface and state of interconnection. A interconnection can acquire these states: Upstream, Downstream, Offer, Waiting, Rejected and Error. If we create offer to remote CDNi, state will change to "waiting" and we have to wait on acceptation or rejection offer by administrator of remote CDNi. Then administrator of remote CDNi can add new entry between interconnection with state "offer". He can accept offer or reject. If offer will be accepted then our CDN interface will become Downstream CDN and state will be "downstream" and remote CDN interface will become Upstream CDN and value of state will be "upstream". Operation with CDNi interconnections is shown in the end of every entry (line) of table. If some errors occured during creation of CDNi interconnection, state of interconnection will be "error" and we can see Error that occurred in Logs. If interconnection is successfully created we can see updated table of footprints, table with Content Origins and of course table of interconnections.

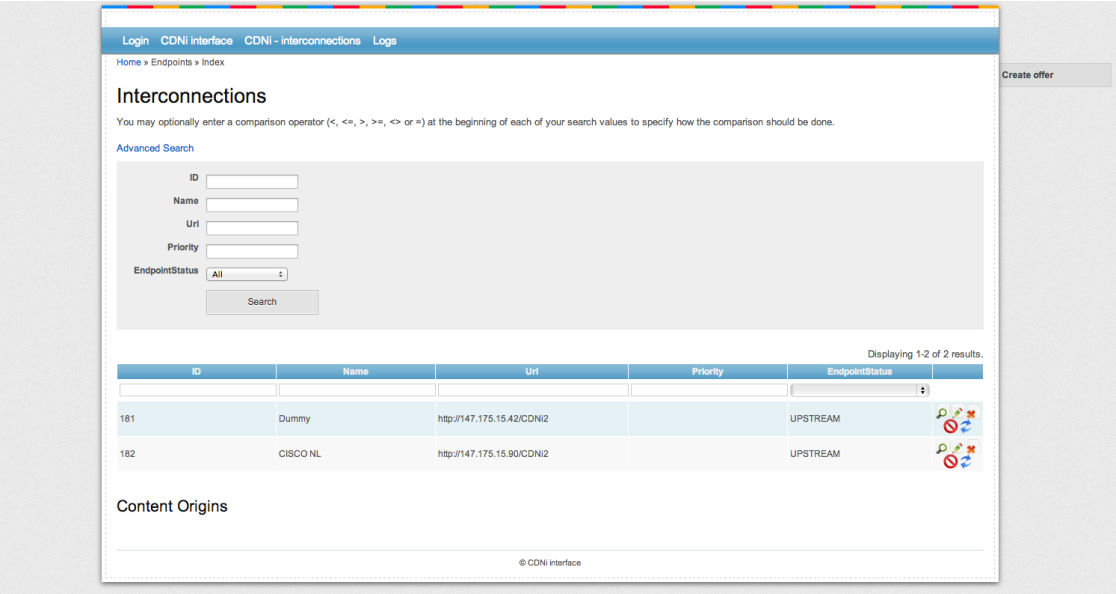

Figure 61: *Page with interconnections.*

We can check all action of CDN interface after click on "Log" item in Top menu. Log view contains logs and administrator can check behavior of CDN interface and in special case detect possible errors. Otherwise administrator can check how CDN interface translates requested URL. If CDNi is gateway for Dummy CDN then administrator can see all URL translations and if CDNi is gateway fo Cisco CDS then administrator can see only request to URL translator and response from URL translator.

For successfully creating of interconnection are needed additional settings in CDS manager. Some of these settings we mentioned in Installation guide and the other are out of scope and it can be founded in official Guide for Cisco Internet Streamer CDS.

C Instalation Guide

In installation guide we describe what is necessary to run CDN interface application. We suppose that server is ready and there is running web server. Our devices were set up with Linux distribution Debian, web server Apache 2 and MySQL database and we recommended to use it too. But, our application contains database layer for creating queries therefore it will be ok to use different database for example PostgreSQL. Web server can be replace too but functionality of module rewrite is needed.

Firstly we create a directory in `/var/www/` that name will be CDNi2. This name is possible to change but subsequently new name you have to change in configuration file of Yii application. In the same directory like is directory where we created CDNi2 we have to create .htaccess file. This file is necessary if CDN interface will be gateway for Cisco CDS. Copy application files that are located on DVD or in repository to created directory (CDNi2). Edit .htaccess file with code from Example 11.

Example 11: *CDN interaface's .htaccess file.*

```
RewriteEngine on
RewriteBase /
RewriteCond %{REQUEST_URI} /CDNTranslationService
RewriteRule (.*) /CDNi2/urlTranslationService [NC,L]
```

From command line on server type command "a2enmod rewrite" to enable module rewrite. For correct functionality of CDN interface we have to create virtual host that will handle URL Translation from Cisco CDS. Open file `/etc/apache2/sites-available/default` and edit it according to Example 12.

Example 12: *CDN interface's virtual hosts.*

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost
        DocumentRoot /var/www
        <Directory />
                Options FollowSymLinks
                AllowOverride all
        </Directory>
        <Directory /var/www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride all
                Order allow,deny
                allow from all
        </Directory>
        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride None
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
```

```
        </Directory>
        ErrorLog ${APACHE_LOG_DIR}/error.log
        LogLevel warn
        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
<VirtualHost *:8080>
        DocumentRoot /var/www
        RewriteLog /tmp/rewrite.log
        RewriteLogLevel 3
        <Directory /var/www>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride all
                Order allow,deny
                allow from all
        </Directory>
</VirtualHost>
```

Created virtual host and .htaccess file helps us to handle request to translation and redirects it to our application. Futher we have to import database. On DVD, we can find exported structure of database. This file we have to import to database using command line or phpMyAdmin. Database name is CDNi2. We can rename it but it is necessary to rename it in configuration file too. After restart of Apache 2 by command "service apache2 restart" will be our application ready to run.

Our application is prototype, therefor we have to set up fully qualified domain name for origin content in application code. Open SoapController.php file located in "/yiiAppFolder/protected/controllers/SoapController.php" and modify function getListFqdn(). We can add as many fqdns as we want for our content delivery network.

In case when CDN interface is gateway to Cisco CDS, we have on mind that fqdn have to be set up in CDS manager and for every used fqdn we have to set up last resort. Setting up last resort is shown on Figure 62.

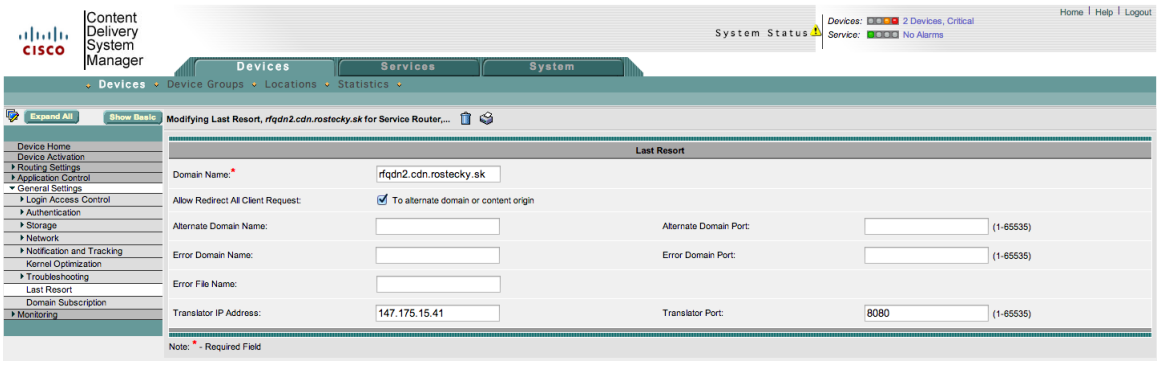

Figure 62: *Setting up last resort on Cisco CDS*

D Resumé

D.1 Úvod

Informačným technológiam sa v dnešnej dobe nevyhneme už v žiadnom odvetví. Každý jeden človek má pri sebe aspoň jeden smartfón alebo prenosný počítač,prípadne televízor, ktorý má pripojený na internet alebo chladničku, ktorá mu napovie čo jej chýba. Informačné technológie sa nám snažia zjednodušiť svet a bytie v ňom. Naša diplomová práca sa zameriava skôr na odvetvia multimediálnej zábavy. Ľudia svoje zariadenia používajú okrem práce hlavne na zábavu a komunikáciu. Radi si posielajú rôzne fotky, pozerajú videá alebo počúvajú hudbu. V nie dávnej dobe by nikoho nenapadlo, že namiesto hudby priamo uloženej v telefóne si hudbu budeme prehrávať cez internet. Spočiatku sme sa tešili tomu, že máme hudbu v telefóne, neskôr boli vysoké ceny za dátové pripojenie, ktoré bolo ešte aj pomalé a teraz má bezmála každý smartfón aj pripojenie na internet. Naše domáce multimediálne zariadenia dostali tiež pripojenie na internet, takže nie je žiadny problem si sputiť video zo svojho YouTube kanálu z pohodlia svojej obývacej izby. Presne tu vznikla idea na vytvorenie služby, ktorá nám zvýši pohodlie sledovania multimediálneho obsahu, ale hlavne nám bude zvyšovať kvalitu doručenia multimediálneho obsahu.

Vytvorili sme zadanie, ktoré malo za úlohu navrhnúť a implementovať službu pre sledovanie multimediálneho obsahu so zaručenou kvalitou doručovania multimediálneho obsahu do Samsung Smart TV. Po úvodnej analýze sme usúdili, že Samsung Smart TV nie je tá správna platforma, do ktorej by sme chceli investovať náš následný vývoj z dôvodu jeho nie zrovna ideálneho presadenia sa na trhu, ktoré bolo spôsobené z nášho pohľadu aj nevhodným používateľským rozhraním. Tu sa otvoril pohľad na Smart TV Boxy od Androidu. Je to otvorená platforma, ktorá je rozšírená a naša služba by bola nielen pre televíziu, ale aj pre mobilné zariadenia. Tu sme sa následne rozhodli, že služba bude vytvorená pre platformu Android.

Pre doručovanie multimediálneho obsahu sme mali zo zadania použiť CDN siete (siete na doručovanie obsahu). K dispozícii sme mali v laboratórnych podmienkach Cisco Internet Streamer CDS, prostredníctvom ktorého mala byť naša služba doručovaná. Vzhľadom na to, že v tomto smere sme nevideli v našej práci vo výsledku žiadny hodnotný prínos , začali sme hľadať spôsob akým by sme posunuli našu prácu na vyššiu úroveň. V rámci celej problematiky sme sa dostali hlbšie a videli sme možnosť efektívnejšie doručovať

obsah prostredníctvom viacerých CDN sietí prepojenými medzi sebou. Tu nastal zlom v smerovaní diplomovej práce, ktorá sa začala zaoberať hlbšou problematikou doručovania obsahu. Vzhľadom na rozsah danej veci sme zmenili zadanie a začali analyzovať prepojenie CDN sietí. Na základe existujúceho ETSI dokumentu, ktorý popisuje problematiku a možné riešenie, navrhli sme CDN adaptér, ktorý bude prepájať CDN siete. Podstatou bolo vytvoriť CDN adaptér tak, aby dokázal prepojiť CDN siete od rôznych výrobcov a aby nebolo potrebné zasahovať do existujúcej štruktúry.

D.2 Analýza

V analýze sme sa na začiatku venovali platforme Samsung Smart TV, ktorá nám vyplývala z pôvodného zadania. Tu sme sa zamerali na všeobcný popis produktu a následne sme sa pustili do popisu platformy. V ňom sme popísali softvérové vrstvy systému a následne sme si aplikačnú vrstvu popísali detailne. Tu sme zisťovali aké typy aplikácií je možné vytvoriť pre túto platformu, aké sú podporované technológie a aké multimediálne formáty sa môžu použiť. Tiež sme tu popísali niekoľko technológií na streamovanie videa.

Druhá časť analýzy bola venovaná sieťam na doručovanie obsahu. Tu sme sa zamerali na vysvetlenie, čo sú vlastne siete na doručovanie obsahu, z akých častí pozostávajú a ako fungujú. Siete na doručovanie obsahu vznikli hlavne pre zvýšenie výkonu pri doručovaní dát a to tým, že sú schopné rozložiť záťaž na viac koncových bodov. Základné uzly CDN siete sú smerovač požiadaviek od klientov, ľubovoľný počet serverov, ktoré vlastnia obsah po ktorom sa používatelia dopytujú a okrajové servery. Koncepcia CDN sietí je založená hlavne na geografickej polohe jednotlivých uzlov a to hlavne okrajových serverov. Komunikácia od klienta sa dedikuje zo servera s obsahom na smerovač. Smerovač rozhoduje, ktorý okrajový server obslúži používateľa. Smerovač sa rozhoduje na základe viacerých parametrov ako napríklad geografická poloha, vyťaženie okrajových serverov, typ obsahu aký je vyžadovaný a podobne. Keď sa smerovač rozhodne kto obslúži daného používateľa, tak ho presmeruje na daný okrajový server. Tento server dostane požiadavku na nejaký obsah, ktorý môže , ale aj nemusí mať u seba uložený. Pokiaľ ho nemá, tak sa spýta pôvodného serveru na obsah a od neho si ho stiahne a následne doručí používateľovi. Takýmto spôsobom je možné vo veľkej miere zvyšovať výkon pri dopytovaní sa po obsahu a jeho následom doručení.

V celom kontexte sietí na doručovanie obsahu sme nemohli v analýze vynechať aj vyriešenie DNS dopytov a ich následné presmerovanie. Pre CDN siete je to veľmi dôležitý

aspekt, pretože na základe doménových mien je CDN schopná obslúžiť dopyt od používateľa a vie na ktorom serveri je daný obsah uložený.

Ohľadom CDN sietí sme v našej analýze uviedli aj Cisco Internet Steamer CDS s ktorým sme v našej diplomovej práci pracovali. Cisco CDS je proprietárny systém od spoločnosti Cisco Systems ,Inc, ktorý so sebou nesie mierne odlišnosti od CDN sietí. Rozdiel je v prvom rade v názvoch jednotlivých uzlov , čo sa vzťahuje aj na ich označovanie. Smerovač na spracovanie požiadaviek od používateľov nesie namiesto "Request Router" názov "Service Router". Okrajové servery, ktoré v konečnom dôsledku obslúžia klienta sa nazývajú namiesto "Edge server" "Service Engine". Cisco CDS obsahuje aj server na menežovanie celej CDN siete a jej všetkých zariadení. Tento server v topológiách nesie označenie CDSM. V Cisco CDS je aj možnosť každému jednému uzlu v topológii nastaviť jeho redundantný uzol, aby v prípade jeho vypadnutia mohol byť hneď nastavený nový. Aby Cisco CDS mohol nejakému serveru distribuovať jeho obsah, je potrebné pre neho v Cisco CDS nastaviť pôvodný obsah, ktorý obsahuje originálnu doménu a doménu k nemu priradenú, na ktorej ju bude Cisco CDS obsluhovať. Následne je pôvodnému obsahu priradený doručovací servis, ktorý obsahuje dodatočné nastavenia o tom ,ako bude Cisco CDS doručovať daný obsah. Na to, aby bolo jasné komu daný obsah môže byť doručený, sa používajú súbory na pokrytie zóny. V nich sa kontroluje a vyhľadáva používateľ, ktorý zastal dopyt po obsahu.

Ako posledná platforma v analýze je Android. Túto platformu sme do analýzy dali z dôvodu, že naša služba mala byť implementovaná v prostredí Androidu.Tu sme sa tiež podobne ako pri Samsung SMART TV zamerali na podporované typy multimédii.

Na konci analýzy sme sa pozreli na existujúce riešenia pre služby na poskytovanie multimediálneho obsahu a na existujúce riešenie prepojenia CDN sietí od Cisco Systems ,Inc s názvom Cisco VDS Service Broker. Nevýhodou Service Brokeru je spôsob jeho implementácie do štruktúry systému. Service Broker je nadradený prvok pre CDN siete. To znamená, že požiadavka od klienta ide cez Service Broker a nie priamo na CDN. Service Broker poskytuje tri pracovné režimy, ktorými sú HTTP GET, DNS a webové služby. Pomocou HTTP GET metódy sa štandardne posielajú dopyty, ktoré v sebe nesú buď cestu na požadovaný súbor, alebo manifest súbor obsahujúci zoznam súborov na distribuovanie. DNS pracovný režim pracuje ako autoritatívny DNS server pre doménu, ktorá identifikuje pôvodný server. To sú atribúty na rozhodovanie sa, ktorá CDN sieť sa použije obmedzene. Pracovný režim s webovými službami umožňuje iným aplikáciám komunikáciu s Cisco VDS Service Brokerom.

D.3 Návrh systému

Návrh systému obsahuje aj návrh služby na doručovanie multimediálneho obsahu aj návrh prepojenia CDN sietí. Pre službu sme si zadefinovali jednotlivé vrstvy, ako majú fungovať a čo majú dokázať robiť. Vytvorili sme základný model interakcie aplikácie v zariadení s webovou aplikáciou. Následne sme si definovali základný model databázy.

Ako sme uviedli vyššie, tak v druhej časti diplomového projektu sme sa zamerali hlbšie na prepojenie CDN sietí. Na základe ETSI TS 102 990 dokumentu, ktorý prezentuje myšlienku prepojenia dvoch CDN sietí, sme definovali návrh CDN adaptéra. CDN adaptér pozostáva z dvoch CDN rozhraní, ktoré sú vstupné brány do CDN sietií. Tieto rozhrania medzi sebou komunikujú a vymieňajú si dáta potrebné na ovládanie CDN sietí. CDN rozhranie pozostáva z troch vrstiev, nad ktorými je ovládacia vrstva. Prvá vrstva zabezpečuje výmenu IP sietí, výmenu možností, ktorými CDN sieť disponuje, výmenu statusov a zaznamenávanie stavu rozhrania. Druhá vrstva sa stará o smerovanie požiadaviek, o výmenu metadát k obsahom a o status samotného obsahu. Tretia vrstva má za úlohu prenášanie obsahu a distribúciu obsahu. Ovládacia vrstva poskytuje administrátorom systému spravovať CDN rozhranie, cez ktoré vie sledovať jeho aktuálny stav, čo vykonáva a tiež prostredníctvom neho vytvárať prepojenia medzi sieťami. Tieto prepojenia sú vždy jednosmerné a definujú, ktorá CDN sieť je nadradená a ktorá podradená. To znamená, že nadradená CDN sieť je sprostredkovateľom pôvodného obsahu a podradená CDN sieť je schopná tento obsah distribuovať ďalej.
V návrhu systému sme tiež definovali základný životný cyklus CDN rozhrania.

D.4 Implementácia

V implementácii sme sa zamerali primárne na výber implementačných nástrojov. Všetky vlastné servery, ktoré sme použili ,majú nainštalovanú Linuxovú distribúciu Debian, na ktorej beží webový server Apache2 a databáza MySQL. CDN rozhranie sme sa rozhodli implementovať pomocou PHP programovacieho jazyka. Aby bola naša aplikácia efektívna aj po vývojovej stránke a aby bola aj prehľadná, rozhodli sme sa použiť programovú sadu, ktorou nie sme obmedzení použitou databázou a ktorá je dizajnovaná za použitia MVC (modely, pohľady, kontrolery). Rozhodli sme sa pre PHP programovú sadu Yii. Komunikáciu medzi jednotlivými rozhraniami sme zvolili prostredníctvom webových služieb implementovaných SOAP technológiou. Yii priamo podporuje SOAP, čo bolo tiež kritériom výberu programovej

sady. Ako testovacie multimediálne dáta sme si zvolili HLS stream. Na jeho vytvorenie sme použili nami vytvorený video obsah, ktorý sme najskôr dali do rôznych rozlíšení a následne sme ho spracovali prostredníctvom vývojárskych nástrojov priamo od Apple. S nimi sme dokázali z daného multimediálneho obsahu vytvoriť segmenty opísané v manifest súbore typu .m3u8. Vytvorili sme aj adaptívny streaming z viacerých manifest súborov, ktorý na základe priepustnosti siete prispôsobí rozlíšenie multimediálneho obsahu.

Pri implementácii CDN rozhrania sme sa zamerali na niekoľko najdôležitejších častí zo zadefinovanej funkcionality tak, aby sme boli schopný vytvoriť spojenie medzi CDN sieťami a úspešne distribuovať obsah naprieč dvom CDN sieťam. Náš CDN interface je webová aplikácia, ktorá je úplne totožná na všetkých rozhraniach. Jediné, čo ovplyvňuje jej správanie je nastavenie atribútov v databáze. Databáza nesie všetky informácie týkajúce sa pripojenej CDN siete, všetky siete, ktoré dokáže obslúžiť, možnosti CDN siete, ktoré sme špecifikovali na typy obsahu, ktorý je CDN sieť schopná doručovať, avšak sú variabilné na akékoľvek možnosti. Základný stavebný prvok aplikácie je koncový bod. Dvoma koncovými bodmi je popísané prepojenie dvoch sietí. Koncové body sa delia na lokálne a vzdialené. Lokálny koncový bod môže byť vždy práve jeden, pretože je to samotné rozhranie pre vstup do CDN siete. Koncový bod je špecifikovaný názvom a URL adresou na ktorej pracuje. Nasledujú doplňujúce informácie, ktoré sú potrebné pre daný typ CDN siete. Pre Cisco CDS sú to napríklad URL a port na ktorom komunikujeme s API rozhraním Cisco CDS ,alebo napríklad autentifikačné dáta, ktorými sa overujeme voči Cisco CDS.

Ako sme už naznačili, komunikácia s Cisco CDS prebieha prostredníctvom API. V čase vývoja sme sa držali verzii 3.2. Aktuálne je už aj verzia 3.3 avšak doposiaľ použité služby neboli zmenené a sú aj spätne kompatibilné aj s verziou 3.1.

Špeciálnou vlastnosťou nášho CDN rozhrania je, že dokáže simulovať správanie CDN siete. Nazvali sme ju Dummy CDN. Správanie sme implementovali tak, že vyhodnotí náležitú požiadavku od klienta, rozhodne sa, kto obslúži klienta, či ona sama alebo iná CDN a pokiaľ ona sama ,tak klienta presmeruje priamo na pôvodný server.

Každé CDN rozhranie sa môže dostať do stavu, že jeho CDN sieť bude nadradená alebo podradená. Nadradená CDN sieť má priamy prístup k pôvodnému serveru a podradená CDN sieť je schopná prostredníctvom nadradenej doručovať obsah. Podľa týchto dvoch stavov rozhranie nastavuje CDN tak, aby bola schopná kooperovať s druhou CDN. Pre nadradenú

CDN sieť bolo treba vyriešiť ako sme schopní donútiť Cisco CDS spraviť rozhodnutie, aby poslala danú požiadavku na inú URL adresu. Tu sme si spočiatku vybrali CDN Selector file. Táto funkcionalita je len v rannom štádiu a len v testovacej prevádzke. Po jej implementovaní sme narazili na problém, že síce Cisco CDS presmerovala požiadavku na druhú CDN sieť, ale nevedeli sme z požiadavky definovať o aký pôvodný server sa jedná. A mala ešte jednu nevýhodu a tou bola nemožnosť rozhodovať sa na základe Ip adries sietí ,ale len na základe rozhodnutia geolokačného serveru. Tak sme našli riešenie v podaní funkcionality, ktorá sa volá Last Resort. Tu sa CDN sieť pri Ip adrese siete, ktorý nepozná, pýta prekladača adries tretích strán. Tento prekladač adries sme implementovali do nášho CDN rozhrania. Komunikácia medzi nimi prebieha prostredníctvom webovej služby SOAP. Naopak v stave podradenej CDN siete naše rozhranie vytvorí v Cisco CDS záznam o novom pôvodnom serveri, vytvorí mu doručovaciu službu a priradí jej Service Engine, ktorý sa o danú službu bude starať. Pri Dummy CDN je to podobné, avšak tieto informácie si združuje v asociatívnom poli, z ktorého si ich číta a vie, ako má spracovať dané požiadavky.

D.5 Testovanie

Verifikáciu riešenia sme spojili s testovaním. Verifikácia mala preukázať, že naše rozhranie je schopné smerovať požiadavky od klientov naprieč rôznym CDN sieťam. Na vytvorenie topológie sme mali k dispozícii 2x Cisco Internet Streamer CDS jeden umiestnený v Bratislave a druhý umiestnený v Holandsku a našu implementáciu Dummy CDN tiež umiestnenú v Bratislave. Pripravili sme si servery s testovacím obsahom a naše CDN rozhrania pre každú CDN sieť. Tieto zariadenia boli tiež umiestnené v Bratislave. Testy, ktoré sme robili sa zamerali hlavne na zistenie oneskorenia aké vnáša do komunikácie URL prekladač adries. Merali sme oneskorenia DNS žiadostí, obyčajné žiadosti s odpoveďou presmerovania (HTTP 302) a žiadosti obsahujúce preklad adries s odpoveďou presmerovania (HTTP 302). Odmerali sme čas potrebný na dosiahnutie prvej žiadosti na samotný obsah ,aby sme do merania nevnášali časy sťahovania objemných súborov.

Výsledkom testov boli v prvom rade časy. Čas potrebný na dosiahnutie prvej žiadosti o obsah používateľovi, ktorého obslúžila nadradená CDN sieť trval pri Cisco CDS v Bratislave v priemere 98ms a bez DNS dopytov 21ms. Pri Cisco CDS v Holandsku to bolo 190ms a 134ms. Používateľ, ktorého obslúžila podradená CDN sieť mal oneskorenie 563ms s DNS dopytmi a 484ms bez DNS. Tieto hodnoty boli pre používateľa pri sťahovaní obsahu patriacemu Cisco CDS v Bratislave. Obsah patriaci Holandskej Cisco CDS sa začal u používateľa sťahovať

po 406ms s DNS a bez neho po 299ms. Tieto dopyty už obsahovali preklad adries. Všetky dopyty na server s prekladom adries trvali v priemere 167ms. Dopyty, ktoré obslúžila priamo CDN a ich odpoveď bolo presmerovanie, trvali v priemere 13ms. Dopyty na preklad DNS trvali v priemere 35ms.

D.6 Zhodnotenie

Naša práca mala byť pôvodne zameraná na vytvorenie služby, ktorá sprostredkuje multimediálny obsah používateľom so zaručenou kvalitou doručovania, ktoré malo byť dosiahnuté prostredníctvom CDN sietí. V rámci práce sme sa dostali hlboko do problematiky CDN sietí a tak sme sa v rámci celého zadania začali na ne viac zameriavať a riešiť práve zvýšenie kvality doručovania. Vytvorili sme prepojenie dvoch CDN sietí prostredníctvom adaptéra pozostávajúceho z dvoch rozhraní, ktoré sú vstupné brány do CDN siete. Riešenie sme verifikovali a urobili sme s ním sadu testov na zistenie efektivity. Výsledky hodnotíme pozitívne a vidíme v tomto riešení perspektívu do budúcna.

E Conclusion of the time plan

Our time plan on third part of master thesis was a little bit changed. Changes were in fifth and sixth week where were focused on mobile application. This week we spent on implementing CDNi interface and testing our implementation. Mobile application was out of scope in this part of implementation.

Time plan Diploma project 3

1. week: Implementation CDNi management
2. week: Implementation CDNi interface
3. week: Implementation CDNi interface
4. week: Implementation CDNi interface
5. week: Implementation Mobile application => implementation CDNi interface
6. week: Implementation Mobile application => Debuging and Testing system
7. week: Debuging and Testing system
8. week: Debuging and Testing system
9. week: Finalize document

F Contents of electronic medium

- cdni folder: Yii application of CDN interface
- database folder: dump of database
- document folder: pdf version of documentation
- tests folder: pcap files from testing
- tree.txt: complet tree of files on DVD