UPTEC IT 11 009
Examensarbete 30 hp
Juni 2011

Event-driven interactivity in IPTV

Fredrik Höglin

UPPSALA
UNIVERSITET

Teknisk- naturvetenskaplig fakultet
UTH-enheten

Besöksadress:
Ångströmlaboratoriet
Lägerhyddsvägen 1
Hus 4, Plan 0

Postadress:
Box 536
751 21 Uppsala

Telefon:
018 – 471 30 03

Telefax:
018 – 471 30 00

Hemsida:
http://www.teknat.uu.se/student

Abstract

Event-driven interactivity in IPTV

Fredrik Höglin

As broadcast television has developed from analogue transmission over the air to digital transmission over cable and satellite, and today over IP networks, many new challenges as well as opportunities to develop new services has surfaced.

This thesis will provide an overview of the IPTV technology with a focus on interactive services, as well as describe a prototype that has been developed to illustrate how interactivity could be implemented. It was observed that sports broadcasting was a good candidate to present additional interactive content, due to the wealth of material usually available outside of the broadcast itself, such as statistics.

A prototype has been developed at TeliaSonera, an IPTV provider with roughly 400.000 customers, as a part of the Next Generation Media project at the Royal Institute of Technology (KTH). The prototype has been integrated into the existing user interface to provide management of sports highlights, enabling users to view highlights in live matches on several channels at once, without worrying about missing any of the action, even if it happens on a channel they are not currently watching.

Interest has also been shown from a third party, using the developed framework for delivering events to the set-top box, and the existing code was modified to allow various types of events to be received, allowing the user to interact with the television content in different ways.

Handledare: Annika Kilegran
Ämnesgranskare: Mats Daniels
Examinator: Anders Jansson
ISSN: 1401-5749, UPTEC IT11 009
Sponsor: Next Generation Media Project

Tryckt av: Reprocentralen ITC

Sammanfattning

IPTV är ett relativt nytt sätt att leverera TV-innehåll till konsumenter. Det skiljer sig från andra sätt att leverera TV-signaler genom att leveransen sker över ett managerat paketförmedlat nät som kvalitetssäkrar signalen till användaren. Till skillnad från analog och digital-TV så behövs ingen tuner för att ta emot signalen, vilket innebär att IPTV inte är begränsat av hårdvaran vad gäller antalet samtidiga strömmar som kan erhållas. Begränsningen är istället bandbredden till mottagaren. För program i vanlig upplösning är bandbreddskravet ungefär 4Mbit/sek och ungefär 8Mbit/sek för högupplöst material. Rent teoretiskt innebär detta att det går att ta emot många fler samtidiga strömmar än exempelvis en digital-TV-box klarar av. I takt med att hårdvaran i mottagaren förbättras så förbättras också möjligheterna att göra fler saker än att endast ta emot TV-strömmar.

När det gäller interaktiv television så har det gjorts försök tidigare, redan på 1950-talet så gjordes ett barnprogram där det var tänkt att barnen skulle rita på en plastfilm som var uppsatt på TV-apparaten. Nya tjänster såsom text-TV dök sedan upp under 1970-talet och försök gjordes med interaktiva TV-program. Dessa program hade dock bara en bråkdel av den budget traditionella program erhöll, och mottagandet hos publiken blev således inte så varmt. Amerikanska företag såsom Warner Cable gjorde försök med sin tjänst Qube i slutet av 1970-talet och början på 1980-talet, och trots att tjänsten hade problem bidrog satsningen ändå till att utveckla interaktiva tjänster såsom Video on Demand (VoD) och Pay-per-view.

Trots att interaktiv TV inte är ett nytt begrepp så är det intressant att undersöka möjligheterna som IPTV medför, framförallt eftersom en viktig egenskap för IPTV är att det alltid finns tvåvägs-kommunikation tillgängligt i och med det sätt tekniken fungerar.

Vincent Grosso, tidigare analytiker på AT&T menar att interaktiva TV-tjänster kan innehålla element ur en eller flera av fyra olika kategorier:

- Information - den elektroniska programguiden i de flesta digital-TV-tjänster har en sådan, där användaren kan navigera ett grafiskt gränssnitt och enkelt få en överblick över vad som visas just nu och i framtiden.
- Spel - användare kan tävla mot andra användare i exempelvis frågesport.
- Transaktioner - hyra eller köpa film eller TV-program med fjärrkontrollen.
- Kommunikation - diskutera TV-innehåll med vänner och bekanta via TV:n.

Paralleller kan dras till internet cirka 1994, webbplatser hade precis börjat dyka upp och var ofta ganska primitiva och simpla i sitt utseende, och erbjöd inte särskilt mycket interaktivitet. Oftast bestod de enbart utav text och hyperlänkar. Om vi liknar detta med dagens text-TV, och jämför dagens websidor med all interaktivitet som erbjuds kanske vi kan få en fingervisning av vad som väntar i framtiden för interaktivitet i TV:n.

För att illustrera hur en interaktiv tjänst kan adderas till en befintlig IPTV-lösning har en prototyp tagits fram i samråd med IPTV-experter på TeliaSonera. Prototypen är tänkt att visa skillnader mellan kapabilitet hos IPTV-system jämfört med andra digital-TV eller analoga system. En bra kandidat till prototyp visade sig vara direktsänd sport. Ofta pågår flera matcher samtidigt i en liga, exempelvis Elitserien. Om en användare är intresserad av att följa flera matcher som sänds samtidigt riskerar denne att missa händelser som sker i andra matcher än den som visas på TV:n.

För att lösa detta problem skapas en tjänst som notifierar användaren om vad som skett i andra matcher än den som visas, och användaren kan då välja att hoppa till den aktuella matchen, fast skiftat bakåt i tiden, för att kunna se händelsen. Sättet denna information transporteras är genom så kallade *"events"*, meddelanden som innehåller JSON-objekt. JSON-objektet i sin tur är ett dataformat som är lätt att arbeta med och innehåller data i form av namn/värde-par.

Meddelanden kan skickas antingen via multicast, vilket är ett bra sätt om samma information ska skickas till många användare, eller unicast när endast en användare efterfrågar informationen. Under utvecklingen har båda sätten använts.

Prototypen består av tre delar, en klient som körs på en set-top-box, en server som skickar data via multicast samt ett operatörsgränssnitt som underlättar skapandet av meddelanden.

Systemet är integrerat med den TeliaSoneras existerande IPTV-plattform och fungerar genom att använda en tidsstämpel och klockor som är synkroniserade med hjälp av NTP-protokollet. När exempelvis ett mål sker så trycker operatören på en knapp i operatörsgränssnittet och en tidsstämpel erhålls från en NTP-server. Efter att all nödvändig information fyllts i av operatören skrivs ett JSON-objekt. När klienten tar emot objektet parsas det och tidsstämpeln kan jämföras med starttiden för det aktuella programmet. Utifrån denna tidsavvikelse justeras tiden ytterligare cirka femton sekunder bakåt i tiden för att tillåta användaren att uppleva hela händelseförloppet.

Med tanke på förutsättningarna får prototypen anses som lyckad. Möjligheterna att införa interaktivitet i det existerande användargränssnittet är goda, och då det bygger på HTML och JavaScript är det särskilt lämpat att addera innehåll som kanske redan finns tillgängligt på nätet. För sport kan detta innebära att det går att skapa statistikinformation som användaren kan navigera med fjärrkontrollen, liknande en websida men anpassat för TV:n.

Under utvecklingens gång visades även intresse från en tredje part, att anpassa klienten för att kunna öppna en extern applikation, som även senare implementerades. Detta innebär att operatören i meddelandet kan skicka med namnet på den applikation som ska öppnas, och användaren kan med hjälp av fjärrkontrollen starta applikationen och sedan interagera med den.

Contents

STB set-top box

EPG Electronic Program Guide

VoD Video on Demand

XML Extensible Markup Language

JSON JavaScript Object Notation

PVR Personal Video Recorder

URI Uniform Resource Identifier

API Application Programming Interface

GUI Graphical User Interface

1 Background

1.1 What is IPTV?

> *IPTV is defined as multimedia services such as television/video/audio/text/graphics/data delivered over IP based networks managed to provide the required level of QoS/QoE, security, interactivity and reliability.[1]*

The difference between traditional analogue and digital television is the method of transmission, instead of being broadcast as terrestrial, satellite or cable television, it is being transmitted in a packet switched network, usually controlled by the operator to provide the quality of service necessary for the viewer to enjoy the service. Since it is delivered over a packet switched network, customers need a decent broadband connection in order to use the service, therefore IPTV services are usually provided by broadband companies, using their existing network to deliver content. Similarly to digital TV, IPTV usually requires a set-top box (STB) that contains necessary tools to decode content. By using multicast for live TV, information can be sent to multiple recipients at the same time, making efficient use of the available bandwidth.

High bandwidth to households is a requirement for IPTV, since a normal stream (assuming MPEG-4 encoding) will demand a bandwidth of about 4Mbit/s for SD content and 8mbit/s for HD content and obviously even higher to receive multiple streams. As the STB increases in power and features, and the consumers migrate towards high definition displays, even more options open up to display information other than video.[2]

When ADSL became commonplace to cover the last mile to the customers home and reaching downstream speeds of 12-24Mbit, and even more when ADSL2 becomes commonplace, it is possible to receive at least one IPTV stream while still having enough bandwidth to use other services on the Internet.

1.2 What is interactive TV?

Interactive TV is somewhat of a late bloomer in our modern information society. It has been around since the 1950s in some form (Winky Dink and You, CBS)[2] and in the 1970s the first set-top box appeared and teletext was invented by engineers at the BBC. In 1977 Warner Cable tested their service, Qube, which was an interactive service offering video on demand and enabled viewers to vote in polls. It was successful with the audience, but failed due to expensive equipment. Since then there has been some experiments with interactive television, notable ones include Time Warner's Full Service Network in 1994, which offered video on demand as well as Internet access and interactive programming. The problem with the service was that the cost of the STB was $10000, which is about 100 times more expensive than todays STBs.[3]

It is not easy to pinpoint exactly what interactive TV is, because there are many variations to it, so it will depend on who you ask. Perhaps some will argue that it is any interaction between the user and the television. In that case, simply changing the channel could be considered to be an interactive task. Maybe interactions with content on a second screen, such as a laptop, tablet or smartphone while watching television can be thought of as interactive television. Since even teletext can be regarded as interactive television, it can at least be agreed upon that the definition is broad. These days, being able to rent or purchase content such as movies on demand, is said to be an example of interactive television. Interactive television analyst Vincent Grosso mentions that interactive services can implement one or more of the following:[3]

- Information
- Gaming
- Transactions
- Communication

Some examples of services using these categories could be as follows:

Information – the basic Electronic Program Guide (EPG) in most digital television services is used to help the user navigate the content, providing more information about the programmes, not only timetables, but descriptions of the content as well, and perhaps in the future also related content, as providers start using metadata to categorize their increasing amounts of content.

Gaming – We could imagine a family watching a game show such as "Who wants to be a millionaire" and using their remote control to answer the questions at home, and see how well they perform compared to the contestants, or even the other TV customers.

Transactions – When purchasing content within the TV service, such as video or movies on demand, transactions take place. This could also include transactions for gambling on sports and other events.

Communication – Perhaps you want to recommend some content to a friend, sending them a message that also contains a link to the content you recommend. You could also watch a football game (or other programming) together, in separate locations, and discuss it through the television service.

It is easy to reminisce about the early days of the Internet, when websites were mostly text and hyperlinks, and not very interactive, and compare this to the old teletext-services offered by TV-channels. Eventually websites evolved and incorporated more rich content, such as images and video, as well as interactive components. Interactive services for digital TV evolved as well, also adding on more of these features. If we look at what is available today in IPTV services, we see that almost anything you can access online is available, the difference

is that it is usually made available inside a walled garden, a portal where the IPTV provider can control what the user can and cannot access.

A necessity for an interactive IPTV service is the presence of a back-channel, which will allow an STB to transmit as well as receive information.This is also what distinguishes IPTV from satellite or digital TV, the back-channel is always present, whereas with digital TV it is optional. With the back-channel, it is possible for users to request content, not just receive it.[4]

1.3 Where are we at right now?

The situation regarding interactive IPTV is a complicated one, with content providers wanting to add extra information to their programmes and service providers need a way to decode and present this information. There is a need for a good method to enter all this information, the service provider would prefer to receive the information in the same way from several content providers, and content providers would like to only enter the information once and be able to provide it to different service providers. There are several organizations that strive to provide a standard platform for future IPTV services, such as YouView[1], HbbTV[2] to name a few.

At TeliaSonera there has been demonstrations of interactive services done several years ago, but since then not much has happened. Other service providers employ varying states of interactivity in their television services, for example the BBC with their Red Button-service that provides news updates, weather information and sports results as well as various other interactive features.[5]

Something that has really blossomed in the last two years on various devices is the concept of "app stores", where customers can download specific applications to their computer, smartphone, TV or other device.[6]

There are several TV manufacturers that provide this functionality and/or the ability to view content over the Internet in their TV sets,[3], thus taking advantage of the recently implemented Internet connectivity options of new television sets. It is of particular interest for the service providers to establish a platform within their own systems, otherwise someone else will. Perhaps not in the set-top box itself, but on the television itself or in a second box, for example the Google TV or a Blu-ray player. All of the current generation video game consoles can stream content such as movies, TV-shows and music videos through specific applications. Some content providers are trying to stall for time while they get their own applications out by blocking these devices from accessing their content.[7]

YouView is in a precarious position at the moment, it was supposed to launch in 2010, but delays have made it seem more likely to launch some time in 2012.

[1] http://www.youview.com/
[2] http://www.hbbtv.org/
[3] Philips Net TV, LG NetCast, Samsung Internet@TV, Sony Bravia Internet Video, etc.

It is obviously an ambitious project, with lots of resources behind it, but this does not matter if it keeps being delayed since other technologies will emerge and take its place.

2 Theory

In order to discern what type of interactive service should be developed, we need to look at what is going on in the world regarding interactive television. This chapter also discusses different ways of implementing such a service, and what parameters to consider. Finally some examples of possible scenarios are discussed using use cases.

2.1 What is happening around the world regarding interactive TV?

Several companies have services of varying grade of interactivity, mostly these services are evolutions of the old teletext services. Since the switch over to digital broadcasting, these services have evolved and can now present more content, and offer limited options for interactivity. Examples of such services are Showtime Interactive TV where users can tune in to a specific interactive broadcast and press a button on their remote to access more information about the event and vote on polls. The service uses EBIF (Enhanced TV Binary Interchange Format) to display widgets on screen that show the selected information. For example, during a boxing match a user can access info about the fighters, their record and statistics, and vote on who they think will win.[8]

Showtime Sports® Interactive is currently available on Verizon FiOS. However, Verizon FiOS is not strictly an IPTV service, only Video on Demand (VoD), interactive features and EPG data are delivered with IPTV technology, the majority of the content is delivered on a QAM (Quadrature Amplitude Modulation) channel via fiber to the customers home, where it is demodulated by the ONT (Optical Network Terminal) and turned into an RF signal in order to transport it around the home in a coaxial network.[9]

BBC has the Red Button-service, which provides interactive content for many programmes, averaging 12.7 million users every week, and out of those, 5 million do not use any other BBC interactive services. As IPTV emerges, it is being discussed how it could enhance the existing interactive features of the Red Button-service. Focus is mainly on delivering the BBC iPlayer to IPTV environments, thus enabling viewers to watch the last seven days of TV and radio programming whenever they please, similar to SVT Play[4] in Sweden. Currently, BBC Red Button is limited on digital television platforms by the available broadcast capacity and the capabilities of the device on which it is

[4] http://svtplay.se/

displayed. The different platforms that can receive BBC Red Button range from Freeview, a free-to-air platform, to Virgin Media. This means that not all features are available on all platforms.[10]

If devices which already have strong interactive elements in place, such as the Xbox 360, suddenly start providing TV services, other companies could see themselves being really far behind as far as social interactivity goes, almost overnight, due to the estimated install base of 45 million units worldwide.[11]

The Xbox 360 already has components in place to exchange messages with friends and voice chat which support social interactivity. Because an Xbox 360 is mainly used for games, it also has a significant edge in hardware compared to cable companies STBs, and thus good potential for more features to be implemented in software. Microsoft has already partnered with AT&T to bring their U-verse IPTV service to the console, but may also be looking at providing an OTT (Over The Top) video service to the console.[12, 13]

There are also various OTT applications for Sony Playstation 3, but they are usually VoD only, not live TV[5]. An exception to this is KT's QOOK TV, which is an IPTV service that runs on Playstation 3 in South Korea.

A Nielsen study carried out in 2010 reveals that people use these consoles to consume video content, both from DVD and Blu-ray as well as VoD and Internet video in addition to playing videogames. With these new services it can only be expected to increase. In the study it was found that 72% of people used their Playstation 3 to watch DVD and/or Blu-ray, 43% for Xbox 360, and around 25% used their console to watch VoD.[14]

2.2 Second screen

An area where there has been little little innovation the past 50 years is the remote control. It has been virtually unchanged since its inception in 1956, when the Zenith Space Command was introduced. Modern remotes have the same basic functionality, plus a few additional buttons. Today's smartphone and rapidly increasing tablet business could change that, however. In the connected home smartphones and tablets share the same wireless network as the typical IPTV STB, making it a trivial task to interact with the STB using one of these devices.

Since the remote control is quite limited when it comes to interactivity, mainly due to the lack of any decent text input method and the fixed nature of its buttons, it is an interesting development to think about what could be achieved using for example a smartphone. The smartphone offers a screen, obviously, as well as decent text input methods that could be used to browse content more easily, by searching. It also has significant processing power, meaning that we

[5]VidZone, MUBI, Qriocity, Netflix, Hulu, ITV, Channel 4 etc.

could offload much of the work required to implement interactive services from the STB where resources are limited.

Today the second screen (in this case smartphone) can already be used in Telia-Sonera's IPTV service to schedule recordings and see what is on TV. Almost every IPTV provider is either currently developing or has already launched their own applications as well.

There are also several applications available that enable smartphones to control media center applications for HTPC's, such as XBMC remote.[6] Verizon, for example, has an application that indeed lets a user remote control their TV service using an Android-smartphone.

With the improved interface a second screen device brings, users could find it easier to discover content in the ever increasing libraries of their IPTV services, as well as making interaction with content easier. Since a smartphone is often integrated in the users social life (Facebook, contacts) it could possibly bridge the gap and allow users to socially interact around IPTV content as well, without increasing the requirements of the IPTV client too much.

2.3 How could we implement event-driven interactivity?

First of all, we need some basic building blocks within the IPTV service to facilitate communication. Since IPTV is IP-based, we already have two-way communication, so if we add message handling capabilities to the user interface, we can communicate with the STB. The second thing we need to do is to capture the input from the users, since users are the ones doing the interacting. This input needs to be redirected to the appropriate functions within the client, to navigate the interactive on-screen widgets. Since events are what is supposed to drive the interactivity in this scenario, we also need some way to create these events.

In order to highlight the features of IPTV, we should consider implementing a service that can leverage these capabilities, such as not being locked in a certain frequency range for channels, providing in theory limitless numbers of channels, as well as the opportunity to record all broadcast content in the network, for the users to consume whenever they please.

2.4 Metadata

When a service is rich in content, such as IPTV, it becomes increasingly important to be able to sort and manage all of the information. This is usually done with metadata. Metadata describes the content in such a way as to make it easy to access and browse. For instance, a movie may have metadata regarding the genre, the starring actors, the director, a description or other things that

[6] http://code.google.com/p/android-xbmcremote/

might be relevant. This data can then be used to view all films by the same director for example. It becomes necessary when the user goes from being a passive viewer to an active consumer, searching for content that matches their taste.

The same can be said for interactive events, data that describes the event is necessary in order to classify it and treat it appropriately. An event that occurs when a goal is scored might be treated differently than an event that promotes a product during a specific moment in a television show. This also opens up the possibility to store events, in this case goals scored, for later consumption, categorizing them with help from the metadata so that users can easily find the event they are interested in viewing.

2.4.1 XML

Extensible Markup Language (XML) is a standard designed to describe data, so it would make sense to apply it in the case of metadata for television as well. It is widely used to share information between many different kinds of systems.[15]

The event, in this case, would be an XML document which in turn contains data in a format that is easily understood by the software that describes the event, such as JavaScript Object Notation (JSON)[7].

2.5 Event delivery

In order to notify the viewer that an event has taken place, there needs to be a way to communicate with the STB. This can be achieved in several ways, each with their own advantages and disadvantages. We can also use the connectivity of the STB to only transmit a small piece of information for notification purposes, and then only those interested in the event can request any additional content through the back-channel.

2.5.1 Multicast

In order to deliver content to a large number of devices, multicast is ideal since each packet is only sent once, regardless of how many receivers there may be. The packets are copied at each router, making it very scalable.

2.5.2 Multicast with information retrieval

Another option is to only use the multicast channel for notification about an event, and include information about where to retrieve the necessary data. This

[7] http://www.json.org

data could then be retrieved using a regular HTTP request or something similar. Something to take notice of is if customers STBs are set to automatically respond to these multicast notifications by retrieving data, it could potentially generate a lot of simultaneous requests.

2.5.3 Long polling

When using long polling, the client sends a request to the server, and if the server does not have any new information to reply with, it stores the request until such information is available. Since this is unicast, each packet only reaches one client, and in the scenario where an event coupled with live TV is being requested, it would generate a lot of traffic, every packet needs to be sent to every client simultaneously, making it a less ideal approach for a large scale system such as an IPTV service.

2.6 Content ingestion

In this case content ingestion refers to the way broadcast material is stored for later consumption. It could either be stored in the IPTV network, for users to stream over unicast, or it could be stored locally on a harddrive in the consumers STB. Each option has their advantages and disadvantages.

2.6.1 Local ingestion

Local ingestion means that content is stored locally, thus a user would have to subscribe for events on a specific channel, and record that channel to the STB in order to view the events. The limit is the bandwidth and also the STB itself, it cannot capture many channels at the same time due to these restrictions. It may be likely in the future, with more powerful STBs and more available bandwidth that a customer can capture more channels locally. However, that would also mean that the hardware would be more expensive and that the increased cost would be multiplied by the number of customers. This means that at the moment it is not really an interesting scenario to consider.

2.6.2 Network ingestion

Network ingestion means that the service provider captures content in the network as it is being broadcast, thus enabling customers who do not have powerful STBs with storage capacity to enjoy content in a time shifted mode in addition to normal live content. This can be used in conjunction with, for instance, the goal service prototype to do the necessary time shifting of the content. This means that the network will experience an increase in VoD-traffic, something that is believed to increase regardless in the coming years.[16]

IPTV providers develop new services that take advantage of the IP-technology, such as the ability to re-watch a program that is ongoing, pause live content, and watch programmes that have already finished airing. All of these examples turns what would be a multicast stream (broadcast television) into unicast streams (video on demand).

2.6.3 Live television

Timeshifting of live television is possible with both methods described above, however, in the case of local ingestion only a predetermined channel or the current channel can be timeshifted due to todays bandwidth limitations. With network ingestion, all content is stored in the network, and the user can jump back in time on any channel immediately, which is essential for the kind of prototype that is being described.

2.6.4 On demand content

On demand content is typically already stored on servers in the network. Once a user requests some content, bandwidth for that user is reserved to ensure good quality playback. Technically content could also be downloaded to the consumers STB before playback can begin, but this would require a harddrive or other type of storage associated with the device, and not all STBs can handle this.

2.7 Use cases

The purpose of the following use cases is to describe cases that differentiates IPTV from normal digital television, especially when it comes to interactivity.

2.7.1 Following highlights in sport events

Usually there are several matches being played at once, for instance in ice hockey. A user may be interested in multiple games being played at the same time, thus the need for some sort of notification. With IPTV and the presence of a network Personal Video Recorder (PVR), there is the option of timeshifting live content, so users never miss the action going on in other games. Highlights will also be tagged and stored, so users can access them easily at a later time.

1. User starts watching one of the concurrent matches.
2. A goal is scored in one of the games the user is not watching.
3. An event is created and transmitted by multicast to the STB.

4. System shows a notification on screen.
5. User decides to view the event by pressing `RED` on the remote control.
6. A popup is displayed with detailed information about the event, and a button is highlighted.
7. User presses `OK` to activate the highlighted button.
8. System switches to the channel the event happened on, but timeshifted to about 30 seconds before the highlight happened.
9. Having viewed the event, the user presses `STOP` on the remote control to return to previous channel.

2.7.2 Sharing content with friends

Using the foundation laid out in 2.7.1 we can extend it to allow users to send messages containing descriptions and references to any content, not just a specific time in a sports event. For instance, this could be used from within a second-screen application, such as a smartphone application, to recommend certain content to friends that might be interested.

1. User is navigating the content of the IPTV service on their smartphone device.
2. User selects a program that they think their friend might be interested in.
3. User selects an option to send this program to their friend.
4. The system formats a message containing the name of the sender, the title of the program and a reference to the content.
5. The friend receives a notification from the system that a recommendation is available.
6. Similarly to steps 5-6 in 2.7.1 the user can display information about the notification and choose to view the content by pressing `OK` in the popup.
7. The system starts playback of the recommended content.

2.7.3 Extra statistics and information during an event

This could be useful during half time in soccer for example, to view statistics for players in the current game, or for the entire season for players and teams. It could be easily adapted for different sports or events, not only sports.

1. User sees a notification that interactive content is available for this event.

2. User presses the appropriate button, and is presented with a widget displaying statistics about the teams/players.
3. User can navigate with the arrow keys on the remote inside the widget to select different teams/players/information.
4. User can close the widget by pressing `EXIT` on the remote.

2.7.4 Exploring multiple camera angles during a broadcast

Since there are no real restrictions on frequency spectrum within an IPTV service, it is relatively trivial to broadcast multiple channels, for instance several channels for the same sports broadcast, each showing different camera views, and a channel that is a sort of mosaic view of all the different camera views, enabling the user to easily identify and select the camera they want to view. This could mean that the user could select which corner camera to view during the pause between rounds in a boxing match, for example.

1. User is watching a sports broadcast, for instance a boxing match.
2. User receives a notification that there is an interactive camera mode available.
3. User presses the `RED` on the remote control.
4. System changes channel to the camera mosaic channel.
5. User selects which camera they want to view.
6. System changes channel to the camera angle the user selected.

2.7.5 Simple polls

Users can vote in polls and see how other users voted in "real time". This takes advantage of the two-way communication in the IPTV network by sending the results of what users voted for to a server, and then passing the information back to the client. This would take up little bandwidth and would not be very advanced to implement. It could be used in sports broadcasting, where users could vote for which team they think would win. It could also be used for other types of events where it could be interesting to see what other users think, such as the Eurovision song contest and similar events.

1. User sees an indicator that there is interactive content available.
2. User presses the appropriate button.
3. User is presented with a question and a selection of answer options.
4. User selects an answer.
5. User is presented with the results of how other users voted.

3 Method

The methods used in this thesis are shaped largely by the circumstances in which this project is carried out, mainly at the offices of TeliaSonera Broadband Services Product Development. This influenced the methods to include qualitative discussions with experts on IPTV and user interface design.

3.1 Literature study

In order to get an overview and introduction to the subject numerous research papers and IPTV news sites have been consulted. Not all of these provide relevant information for this report, but they served to build an understanding of the subject. In addition to this, there have been interviews with employees at TeliaSonera Product Development regarding the IPTV back-end, IPTV service and its functionality. This helped shape the prototype, since not all scenarios were realizable.

Due to the nature of the subject, many of the resources have been gathered online, this is mainly due to the rapidly evolving technology, meaning that anything other than general observations about the technology in print is likely to be outdated after a couple of years. Books have been proven useful mostly in describing the background of IPTV services, not so much with regards to current developments.

3.2 Prototype

The assumption is that it is possible to create an attractive interactive application that adds value to the overall IPTV service. To illustrate this, a prototype is developed, since it will provide an understanding even to those who are not aware of the full potential of an IPTV system. The prototype highlights differences between IPTV and digital TV and is designed to be extended to provide a basic interface for future interactive services. Due to time constraints, only one possible prototype is being developed, to illustrate one of the uses of event-based interactivity.

3.3 Development method

For development, the principles of agile software development were applied. These principles help keep the focus on working code, and allow for changing requirements. Since the full extent of the prototype was not known beforehand, it became clear that it would have to be easy to make changes throughout the development process. Due to the implementation of the prototype as part of a larger code base, it was a good fit to first start with integrating a very basic

scaffolding of working code and then add on more features until the desired prototype was achieved. This also had the benefit of having a somewhat working prototype in early stages, for demonstration purposes, and it could be used to receive valuable feedback continuously during the development. This was essential due to the complexity of the existing code base, it would be difficult to first understand everything and then plan accordingly, so instead changing requirements were welcomed as they appeared, together with a growing understanding of the system.

3.4 Use cases

In order to maintain an overview of the capabilities of the prototype, and to make discussion about features easier, use cases were used. Use cases provide a way to structure what the prototype can and cannot do, making sure the prototype performs as specified, provided the implementation is done properly. The use cases are not exhaustive, as they would be for a marketable service, however they cover enough cases for the prototype to serve its purpose.

Use cases describe the system from the user's perspective, and describes the tasks a user would do to accomplish a certain task. As the requirements change, so does the use cases, if it is warranted.[17]

4 Prototype details

From the methods described in 2.6, the idea of using local ingestion can be discarded. It is not really useful for a user to be limited to tracking a single channel, when the content of many channels could instead be stored in the network. Out of the different use cases, it was decided that the sports highlight tracking prototype could be a useful example of the kind of interactivity that IPTV can offer.

The prototype basically consists of three parts:

- STB client to receive events and display information on the TV screen, see figure 1.
- Server that communicates with the STB client.
- Interface for enabling an operator to easily enter events with all relevant information.

One of the first, and probably the greatest obstacle in designing the prototype was to decide on how to implement the communication. At first glance it seemed most appropriate to create a client to run on the set-top box that listened on a multicast channel. Messages would then be sent from a server on another

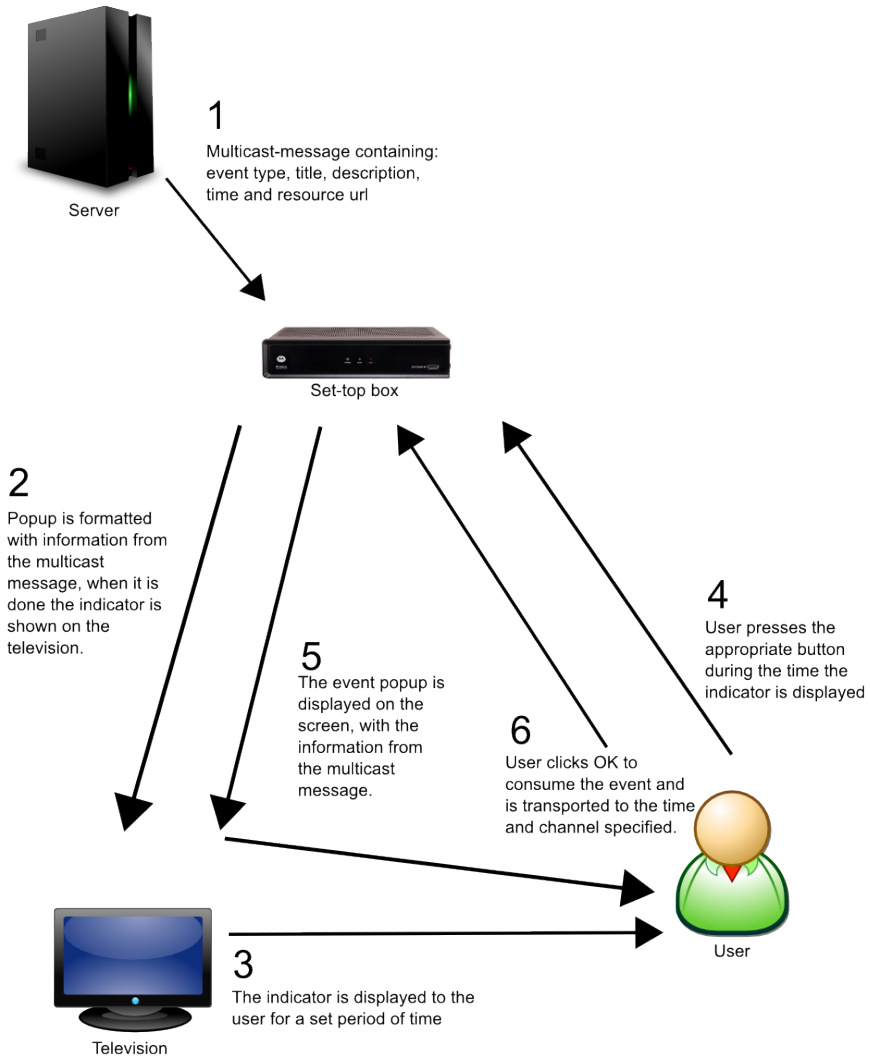

Figure 1: Client overview

machine, and these messages would contain the necessary information to enable the set-top box application to retrieve and display the interactive content.

Due to the fact that it is not optimal to have every STB respond to a multicast message with a request to a server, it was decided that the best course of action would be to embed enough information in the multicast message so that a dialogue could be presented to the user. This would have the benefit of the server receiving the requests over a larger time frame, and not from STBs who are idle. It would also be possible to show the user what kind of event just took place, requiring one less button press to do so. In the case of sports highlight events, all of the required data could be embedded in the multicast message, requiring no additional queries what so ever.

Regarding the client, it could either be implemented as an external application that the user had to start manually whenever they were interested in receiving interactive events, or it could be implemented as part of the user interface, running in the background once enabled in the settings menu, listening for events. To make it easy for the user, and to make it less obtuse to use the service it was decided that it would be implemented as part of the user interface, and that it could be enabled or disabled using a setting in the menu on the STB. In the future this could also make it easy for users to subscribe to various events according to their interests, just by selecting them from the settings menu.

Since events themselves just contain text, i.e. the description of the event as well as the content Uniform Resource Identifier (URI), they do not take up much bandwidth, and keep the memory footprint in the STB low. If graphics such as images are needed, they can be downloaded after the event is received but before the notification is presented to the user.

In the future, if the service is used for many sports simultaneously, it will become necessary to sort highlights perhaps based on sport, league and/or game, in order to make navigation easier. The amount of recent events stored as highlights should depend on the number of games played. Taking examples from the Swedish hockey league *Elitserien*, one day could have 4 simultaneous games, and the amount of goals scored could be for example 27 (round 2, September 18 2010) so some kind of margin is needed if all the recent highlights from the days game should be viewable. Obviously this will vary depending on what kind of sport the service is used for, but it is something that needs to be accounted for nonetheless.

4.1 Detailed use cases

These are use cases that describe the prototype in greater detail.

4.1.1 Basic use case

This is the basic use case described in 2.7.1.

Prerequisite: User has enabled the setting to receive highlight notifications. User is watching live television.

1. User starts watching one of the concurrent matches.
2. A goal is scored in one of the games the user is not watching.
3. An event is created and transmitted by multicast to the STB.
4. System shows a notification on screen.
5. User decides to view the event by pressing `RED` on the remote control.
6. A popup is displayed with detailed information about the event, and a button is highlighted.
7. User presses `OK` to activate the highlighted button.
8. System switches channel to the channel the event happened on, but timeshifted to about 30 seconds before the highlight happened.
9. Having viewed the event, the user presses `STOP` on the remote control to return to previous channel.

4.1.2 Basic use case variant 1

This is an alternative flow from step 4 in use case4.1.1. A notification is received, but the user is not interested in viewing it.

1. System shows a notification on screen.
2. User ignores notification.
3. Notification disappears after a set amount of time.
4. Highlight is stored in the list of recent highlights.

4.1.3 Basic use case variant 2

This is an alternative flow from step 9 in the use case 4.1.1. The user has viewed a highlight, and wants to continue watching the same game.

1. User does not press the `STOP` button after having viewed the highlight.
2. User remains on the same channel.

4.1.4 Basic use case variant 3

This is an alternative flow from step 3 in the use case 4.1.1. In the case of the notification already being displayed while an event is received, the new event should be stored in the queue, and only displayed when the first notification has expired.

1. An event is received while the notification is displayed.
2. The event is stored in the queue.
3. After the notification has expired, the new, not yet shown event is displayed.
4. Continue from step 4 in 4.1.1.

If the user is navigating menus when receiving an event, the notification will not be displayed. This is because it might interfere with the GUI. It could instead be displayed once the user has returned to regular viewing.

4.1.5 Basic use case variant 4

This is an alternative flow from step 3 in the use case 4.1.1.

1. Highlight event is added to the queue.
2. Since user is already watching the channel, the notification is not displayed.

4.1.6 User wants to view recent highlights

1. User presses remote control button to bring up a list of recent highlights.
2. User selects the desired highlight from a list widget on screen.
3. Continue from step 6 in 4.1.1.

Exception: There are no highlights to view.

- The user is presented with a popup stating there are no highlights to view.

4.2 Client

The client is basically a module for the existing GUI, written in JavaScript. The existing code is modified to relay multicast messages that carry interactive event content to this module, where the GUI elements will be created depending on what type of event was received. The underlying framework will also provide a way to forward button presses from the remote control to the module, so users can interact.

Since the STB has very limited hardware resources it is important not to add too many memory consuming elements, but instead use existing ones. This will also help keep the new module consistent with the rest of the user interface, ensuring a uniform experience throughout.

The client needs to receive notifications about events the user is interested in, and present this in a manner that fits neatly into the user interface, without unnecessary distractions. It also needs to present the appropriate content, as specified in the message. For a system with many users such as an IPTV service, what makes the most sense is to use multicast, especially in the case of events on live TV, as each user receives the same exact event. The client could also handle different types of events, since the event itself could have a different type and URI associated with it. Examples of this could be a highlight event for live TV, an event that notifies the user that extra statistics is available for an athlete or a team in a sports event, it could even be a recommendation from a friend containing their name and an identifier specifying that particular content, such as a movie or a television show. It would still be transported in the same way, regardless if it is a sports highlight event or if it is a recommendation event, and any additional content needed to present it such as images, links to content or even an interactive widgets, could be fetched by using the URI.

At present the client has a function called `addEvent` that takes one argument, a JSON-object. This can allow for the client to do different things depending on the `TYPE` of the event, and thus can be extended in the future by other developers. For instance, a polling function could be added to the client which checks for new information, and creates a JSON object based on this information and sends it to `addEvent`.

4.3 Server

The server is a small piece of TeliaSonera-created software running under Windows. The Windows-environment is run on the development machine in a VirtualBox[8], but the server could just as well be run on another machine on the same network. The server can transmit XML data over multicast, and the event-messages are specified within the XML files. A message contains a unique ID, a subject which in this case is "`JSONEvent`", and a body which contains the actual

[8] http://www.virtualbox.org/

JSON object. Furthermore, the JSON object contains different attributes as well, depending on what type of event it is. The JSON syntax is very readable and adaptable, so these could be changed as necessary. Basically for a sports highlight event the JSON object will contain the following:

- `id`, a unique identifier to distinguish between events
- `type`, for instance, "goal" if a goal was scored
- `title` which is a short description of the event for presentation purposes
- `description` which is a longer description of the event, with more information
- `resource` which is a link to the content within the IPTV network, together with the timestamp and information about when that content was recorded, an appropriate time before the event occurred can be calculated.

Depending on the `type` of event the other attributes can be easily changed, since it is possible to use different functions to manipulate the data within the JSON object, and the way to distinguish which function to use can be determined by the `type`.

The `id` attribute needs to be unique for each event, so the client can differentiate between new and old messages. In order to accomplish this, the UNIX timestamp of the event creation time is used. This also serves to calculate when to start playback of a video clip, assuming that the recording start time of the content is known.

The server will broadcast the contents of XML-files using multicast, but will unfortunately not transmit the contents of the files as they are updated. This means that during development there was a noticeable delay between writing an XML-file to actually receiving the message on the STB or in the Firefox-window. This could of course be fixed with access to the source code, by reducing the interval with which the files are read. It is however mostly a matter of convenience and not something that is critical to the function of the prototype.

4.4 Operator interface

The content provider interface is a simple GUI application written in Python. It is used mainly for testing purposes, to get the timestamp in Unix time from a server, since it would be quite tedious to enter this by hand for every message. The program overwrites `Messages.xml` each time an event is created. If this service was to be deployed, or used for different kinds of messages, a better interface would be needed. What the interface looks like can be seen in figure 2.

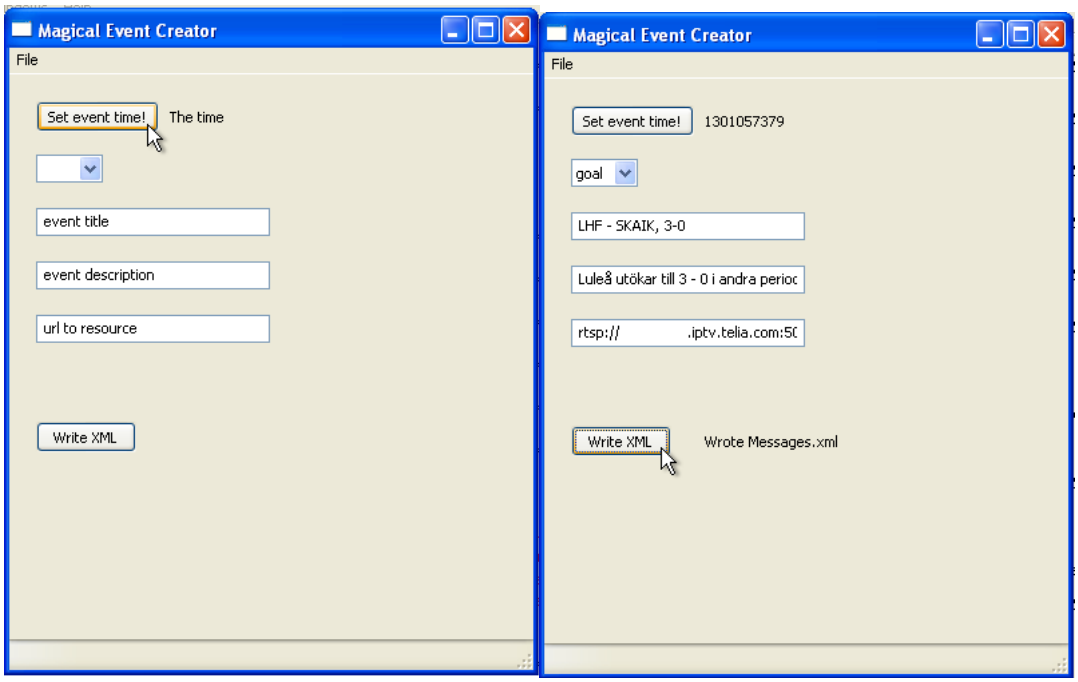

Figure 2: Operator interface before and after creating an event

If, in the future, there is a need for different kinds of events, it is relatively easy to edit the Python source code to allow for this. Just by adding the appropriate edit boxes or list items, and of course updating the function that writes the XML-file, a custom JSON event can be created. These new JSON attributes could then be handled in the JavaScript source code on the client.

4.5 Development

Development is done on a desktop computer running Ubuntu Linux, and a STB from Motorola. In addition to this, since the GUI is written in JavaScript, it is tested in Firefox with the Firebug extension to help with debugging. To emulate the server that provides multicast streams, a VirtualBox instance running Windows XP was set up within the Linux environment. Within the Windows XP install there is a simple multicast server running, that checks to see if any files have been updated, and if so, transmits them to the specified multicast addresses. In order to edit the message file, the operator interface is also run from this virtual box.

4.5.1 JavaScript Module Pattern

When writing a complex piece of software such as the graphical user interface for an IPTV service, it becomes necessary to minimize stability issues and restrict access to parameters that are not essential. As the amount of global variables increase, so does the probability that the code will be unstable or less secure than it could or should be. In JavaScript, a way to remedy this is by hiding things within the closure of a function. The inner function can always access the variables and methods of its outer function, but hides the variables and methods from other functions. A handy design pattern to do this is the JavaScript Module Pattern.

Everything in JavaScript is represented by objects. Objects can have different values, and these values can themselves be objects, functions, arrays et cetera. If a value is a function, it is called a *method*. The method can then access variables that belong to same object. These are accessed by using the `this` keyword.

The JavaScript Module Pattern was proposed by Douglas Crockford, and provides a clever way to have private variables and methods by assigning the return value of an anonymous functions to a namespace object. In JavaScript, if a programmer would forget to declare a variable by omitting the `var`-prefix, it would be implied to be a global variable. In order to minimize the number of global variables, these can be enclosed in an anonymous function so that variables and methods necessary for a module can be accessed within that module, thus avoiding pollution of the global namespace. The object that the anonymous function returns can contain members which are public, but it can also contain members

before the return statement, which are then private. This is possible because of closures. The inner functions can access variables and methods of their outer function. If a function outside a module needs to interact with some variable inside the module, it can do so by calling a public function that operates on the private variables of the module, this is called a privileged function.[18]

4.5.2 Calculating when to start playback

Ideally playback should start early enough before the event so that the full highlight can be appreciated, for example the play leading up to a goal. This means that the internal clock on the STB must be synchronized with the clock of the service that is generating the XML message that is being transmitted. Instead of doing unnecessary work synchronizing these two together, NTP (Network Time Protocol) is used.

When an event is created, the operator presses a button in the operator interface described in 4.4. This button queries an NTP-server and embeds the time in Unix time (the number of seconds since January 1st 1970). The STB also synchronizes with a similar service, so the two clocks can be presumed to be quite well synchronized, at least good enough for the purpose of this prototype.

For testing purposes, a target time of 30 seconds before the goal occurred was decided. This would then be calculated by using the start time of the recording and comparing it to the time received in the message, minus 30 seconds.

As an example, a hockey game that is scheduled to start recording at 19:00 has a goal happen at 19:34:56. The operator would press the button as soon as the goal occurred, giving the message a timestamp close to that time. This message would, after all the other necessary information is entered, be transmitted to the STB. It would the be calculated on the client that playback should start 34 minutes and 26 seconds in, giving the viewer the chance to appreciate the play leading up to the goal. Since we start playback around 30 seconds before the goal, it is not really significant if it differs one second one way or the other, so no more exact measurements other than using a debug printout in the GUI, printing the difference between the time the event was received, and the time embedded in the JSON object. This was observed to only differ a couple of seconds at most.

4.5.3 JavaScript Debugging

During development the bulk of the GUI was done in the Mozilla Firefox browser with the Firebug extension. This allowed for fast loading times of the GUI (around ten seconds, compared to 40-60 seconds on the STB) and easy debugging with Firebug.

Since JavaScript does not need to be compiled before running in the sense that programs coded in C need to pass gcc, it can be harder to detect bugs. Even

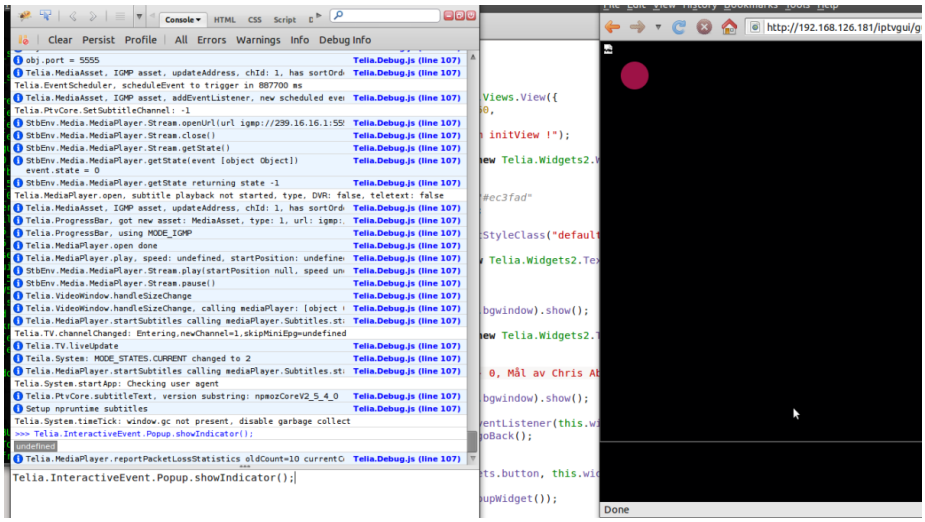

Figure 3: A screenshot of the GUI and the Firebug console during development.

code that is missing semicolons and other errors will not stop the code from being executed, so it can be difficult to detect where it is going wrong. This is where Firebug comes in, it allows you to insert breakpoints anywhere you want in your code, and stepping through each statement, inspecting values as you go. This was immensely helpful in tracking down bugs throughout the development process.

To find minor syntactical errors and to generally keep the code clean and readable, JSLint[9] was used. The version used was the command-line version. It will complain about missed parenthesis, semicolons and if code is not indented properly, and it will save a lot of development time in the long run.

A screenshot of the development environment can be seen in figure 3.

4.6 Prototype evolution

After the first prototype using multicast was completed, interest was shown from a third party to use the features of event management that had been implemented. This was recognized as a good opportunity to test the ability to extend the functionality of the prototype to support third party applications.

Since the prototype is designed in such a way as to allow for changing requirements and quick addition of new features, it does not require a huge time investment to make the necessary changes.

First, changes were made to the JSON object. New attributes were added as shown:

[9] http://www.jslint.com/lint.html

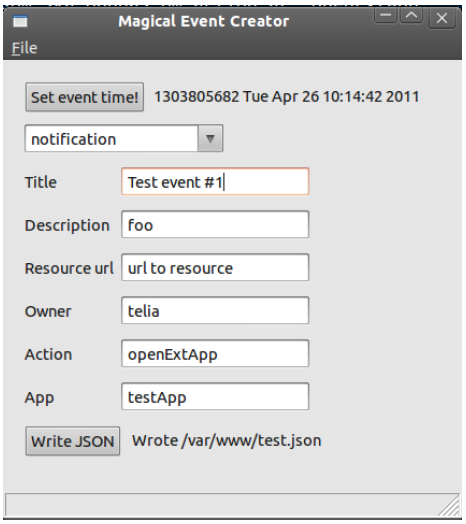

Figure 4: Evolved JSON event writer

- `owner` - for the owner of the event
- `action` - action to be taken
- `app` - application to be launched
- `data` - an object that can contain various types of data, depending on the owners requirements

Since the JSON object is changed, the program that writes these objects has been changed as well, see figure 4.

The old attributes that were specific to the goal highlight service, `resource` and `description` have been moved into the data object, this is done in order to make the format of the JSON object more generic. Any data is required by a specific event type can be added into the data object.

The client was also changed to poll a webserver within a set amount of time to check for updated events. There was code left from the very first prototype that did just that, thus requiring little additional development. This has the advantage of events only taking a couple of seconds to reach the client, after the JSON object has been written, instead of the minutes it would take with the multicast variant.

Furthermore, since the prototype now uses the more rapid event handling that the polling solution allows, there was a need for a queue-system for events, so

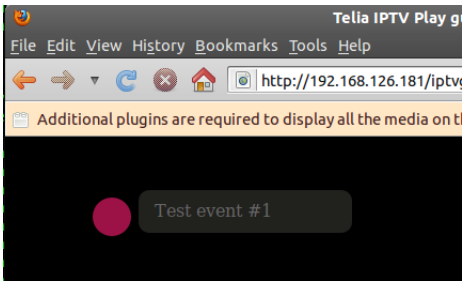

Figure 5: Event notification

that events that are received while another event is displayed can be queued to display their notification after the previous one has disappeared. Another reason for this change was to allow third parties to test the message functionality without requiring the use of a multicast server.

To simulate the launch of a third party application, a simple demo app is used. It offers no functionality except for pressing a few buttons, and is only intended to be a place holder for a real application, illustrating that it is indeed possible to launch an application with information received in the event message.

The latest feature added, at the request of the third party, was to display the title of the event as the notification is displayed, see figure 5.

In the long term, the prototype could be refined further to provide even better support for future third party events, as well as organizing the code a bit more, to allow for easier management of events.

4.7 Prototype evaluation

As the prototype was tailored to meet specific requirements set by the environment at TeliaSonera, and the resources available, it can be argued that it has met the expectations quite well. Some basic assumptions about the system are made, due to the implementation being done in a testing environment where some parameters have to be simplified or emulated.

For instance, there is no access to live captured television, so instead a static video object is used. As a proof of concept, this is enough to show that it is indeed possible to send a message containing a URI via multicast to a set-top box which can then play the content described in that URI at a specific time. The assumption here is that live television can be captured and access via a URI can be provided in the message.

There are also issues with the server that should be pointed out, for instance, the current version of the TeliaSonera-developed multicast server does not transmit new events within a reasonable time frame. This is most likely due to the fact

that the thread that checks to see if new information is available is set to an interval of around two minutes. This is fine for testing purposes, but it should be noted that it simply would not work in any real application of the service. Consider the following scenario; A goal is scored, and an event is created and written to the XML file, ready to be transmitted. Before the multicast server can send the message, another goal is scored, and that event is created and overwrites the previous one before it has been transmitted.

Another, more efficient solution (from a testing standpoint) was the initial way to receive event information, by reading a file from a webserver instead of receiving multicast messages. This means the GUI can do HTTP polling with greater frequency, and receive messages quicker. Since the Python code that is used as an operator interface is platform-independent, it was easily adapted to write JSON-objects to a file on a webserver instead of writing XML-files to the multicast server directory.

The type of information contained in these JSON-objects is also just enough for the prototype to work correctly, and would have to be adjusted accordingly depending on what sort of information the operator wants to manage, it is by no means set in stone and should be revised if it would ever lead to a live service.

Not everything that was planned has been implemented. Due to limitations in the current version of the user interface, it is not possible to resume video on demand-playback after having viewed a goal. This means that the last part of the use case 4.1.1 is not yet implemented. This should also be resolved before a service like this is launched, since it will be quite common for the user to watch some sort of video on demand, and also viewing a goal highlight puts the user in video on demand mode, where the live feed is time shifted, and goes from being multicast traffic to unicast.

All in all the prototype makes good use of existing elements of the TeliaSonera IPTV interface, and presents the necessary information using the same look and feel as the rest of the service. Obviously the graphical choices are not ready for release to the public, but rather the prototype uses the widgets already in the TeliaSonera GUI. The *"OK"*-button in the popup box, for instance, has the text *"->"* on it, which, again, is fine for the purposes of this prototype, but not something that customers should see.

A concern about the design of the prototype is that it is somewhat fragile with regards to how events are created. It is critical that the timestamp is made at the correct time, otherwise the calculation of when to start playback of the highlight will be off, resulting in a useless start time. It can be argued that it would be better to first create the highlight video clips, and then create the event, referencing that specific clip, so it is certain that the correct highlight is displayed.

The memory footprint is also something that is important, but not something that was a priority during development. Due to being unfamiliar with the software beforehand, and the relatively short development phase, focus was

always on producing a working prototype. Thus, the most optimal solutions was not the ones implemented. However, since the prototype is mainly composed of text and existing widgets, the memory footprint is quite low. Debug calls to `memfree` reveal that only around 3-400kB of memory is used when the module is loaded, which is acceptable.

The code, in general, is not as neat and professional as it could have been. This is due to unfamiliarity with JavaScript in general, as most of the coding was done while still learning how to code JavaScript. Valuable resources have been Douglas Crockford's lectures about JavaScript as well as JSLint, to keep the code somewhat organized.

5 Discussion

While there still is no widespread standard for implementing interactive services in IPTV networks, or even IPTV services in general, operators have developed, and continue to develop, their own solutions. What this means for smaller IPTV operators (such as TeliaSonera) is that if they want interactive services, they either have to develop it themselves, which can be difficult due to not being a content owner, as well as costing time and resources. The other option is to convince the content providers to develop these services.

For this to be successful, there needs to be a clearly defined Application Programming Interface (API) to facilitate this. If the development process is too complex for the content provider, they might not bother at all.

Currently, as of April 2011, APIs and other design documents is somewhat lacking for platforms such as YouView, which is unfortunate. Technology is moving quickly, and a project as ambitious and large in scale such as YouView does not benefit from these delays.

The prototype for this thesis was developed on a platform where the Graphical User Interface (GUI) is based on JavaScript, and runs on a Webkit-based browser. The benefit of this is that development time can be rather short, due to the fact that what a third party developer needs to do is to basically make a web page, with some added constraints. These constraints are due to the limited memory and processing power of the STB as well as the difference between a computer monitor and a television screen, and how they are used. For instance, scrolling is not recommended in TV interfaces since it is cumbersome to do with a remote control. Text and icons need to be clearly visible from a greater distance, since most of us sit further away from our televisions than we do from our computer monitors.

The tasks more closely coupled with the service providers interface, such as how these third party applications will be invoked is best handled by the service provider themselves, so that only a limited interface is opened up to the third

party developer, thus preserving the so called "walled garden" within the IPTV platform.

For example, if we consider the National Hockey League (NHL), they have a wealth of information on their website.[10] Information such as team- and player-statistics as well as statistics for each game could possibly be implemented in an IPTV service as well, allowing users to navigate this information on their TV screen during matches, instead of having to use a second device. Technically, this can be achieved with relative ease, but the big challenge is instead a business matter rather than a technical one. The business side of things are not really explored in this thesis, but it is something to consider.

If transactions were to be implemented in the IPTV service, we might see content providers eager to develop their own applications to take advantage of new customers, but this is more of a marketing question than a technical one, and should probably not be discussed in any depth in this thesis.

Sports broadcasting in general lends itself very well to developing interactive services, since there is so much additional information related to the broadcasts, more so than traditional programming. Most sports leagues employ some sort of live reporting on their official website, that fans can follow to get updates on how the game progresses, examples of this are the Swedish top leagues for hockey, soccer and basketball. Users can access all kinds of statistics about players and teams on their respective websites.[11]

Since this information is already presented in a web format, it can be argued that if there was an incentive, it would not be too hard to create a portal within a Webkit-based IPTV service, to display this information alongside the respective games as they are being broadcast. The prototype has shown that this certainly is a possibility.

The design approach, which promoted rapid changes in requirements, have proven to be very useful, since the prototype changed several times during development, but remained in a somewhat functional state almost the whole time, as features were added. It is also light in resources, mostly using text and pre-defined building blocks such as a list widget and text widgets. This means that development time was focused on solving the problem of managing events rather than designing a GUI to fit with the existing service.

5.1 Problems and shortcomings

Due to development being done at TeliaSonera, the demo version of the prototype is not very portable, it depends heavily on the network structure at TeliaSonera's facilities. It is set up in such a way as to re-route many of the locations the television service needs to access, as well as needing the multicast

[10] http://www.nhl.com/

[11] http://www.hockeyligan.se/, http://www.basket.se/ligan, http://www.allsvenskan.se

server in order to even load the user interface. What this means is that only the internal presentation will use a live prototype, and other presentations of this thesis will use recorded video of the prototype.

The user interface is not something that has received a lot of attention during development, instead focus has been on the concept of transmitting and receiving custom events, and displaying the information contained in these events in a simple manner.

The JSON object being written by the custom built Python program has several values that are not yet taken care of in the client. For instance, there is no logic for managing the opening of third party applications, even though there are attributes in the JSON writer that can be set.

Problems also exist when dealing with delayed playback of content, since access to content has been scarce. It has been tested in a limited way, with a delay set manually in the JSON object. The idea has been to not use this hard coded offset, but instead specify a start time for the content, and use the timestamps to calculate when playback should start. For instance, the JSON writer could be extended to fetch a list of sports broadcasts, and their start times, and the operator would then choose the correct game from a list within the interface when creating the event, and the start time of that specific match would be embedded in the JSON object. It remains to be seen what approach will be chosen, if others were to extend the work done in this thesis.

It should also be added that so much has happened during the course of this thesis, that information gathered during the early phase regarding what happens around the world might not be up to date. It should be noted as something that was valid at the time, during autumn 2010. Almost all IPTV companies studied have developed several new services and types of interactive content. Keeping up with all of these developments around the world would leave little time for developing the prototype and writing this thesis. Another thing is that the IPTV business is fiercely competitive, with IPTV providers keeping their developments secret, making it difficult to uncover what companies are developing before they are ready to unveil it publicly.

5.2 Further work and research

First of all, the JavaScript code should be cleaned up a bit. Functionality regarding the receiving of events and the handling of different types of events should perhaps be divided into different files, making it more modular and structured, making it easy to add on more functionality in the future.

The functionality regarding third party applications is implemented, so that it is possible to launch the third party application using the button on the event pop-up. However, there is currently no application other than an empty demo-app to launch.

Something to consider is that interaction does not necessarily have to be done on the STB using the remote control, it could also be done on another device, a second screen. The important thing is that there is interaction around the content displayed on the television. Something that was planned but dropped due to lack of time was to do a survey regarding the availability of second screen devices in the customers homes. However, looking at the increase in smartphone and tablet sales, some tentative conclusions about the possibilities of interactivity with second screen devices are not too far fetched.

A serious issue that needs to be looked into is how to handle the playback of content. This was simulated during development, where the URI of the content, as well as the start time (and in previous iterations the offset) was hard coded. The content is something that is used internally in the IPTV service, and some sort of API for third party developers is necessary in order to provide correct content to accompany these kinds of sport highlight events.

The user interface also needs a lot of attention, but this is left up to the service provider, in this case TeliaSonera, as it is very basic at the moment.

The creation of events could also be a topic of research, some articles that were studied during the early phase of this thesis have described various automated approaches to this, analyzing the crowd noise and the pitch of the commentators to determine whether a highlight has taken place or not.[19]

The automated approach needs to be more mature if it is to be implemented, as it complicates things even more. So until it is consistent enough to provide correct highlights, the preferred approach is to have a human being determine when a highlight should be created.

An organized way of tagging these events with relevant metadata could also be useful, if this was to turn into a live service. Consider the scenario where there is an entire season worth of highlights, in different sports leagues. Users wanting to navigate these events to view highlights would find it difficult unless they were able to sort by sport, league, date, game et cetera. This is when metadata becomes important, and unfortunately it is not something that could be implemented very well in the prototype.

References

[1] ITU-T. IPTV standardization on track say industry experts. http://www.itu.int/, retrieved 2010-11-02.

[2] Gerard O'Driscoll. *Next Generation IPTV Services and Technologies*. John Wiley & Sons, 2008.

[3] John Carey. Interactive television trials and marketplace experiences. 1997.

[4] William Cooper. The interactive television user experience so far. UXTV '08 Proceeding of the 1st international conference on Designing interactive user experiences for TV and video, 2008.

[5] The British Broadcasting Corporation. BBC Red Button. http://www.bbc.co.uk/digital/tv/tv_interactive.shtml, retrieved 2010-11-02.

[6] The Yahoo! Connected TV Team. Yahoo! Connected TV Store is expected to launch in March of 2011. http://www.yctvblog.com/blog/2010/11/18/yahoo-connectedtv-store/, retrieved 2010-11-19.

[7] Sam Schechner and Amir Efrati. Networks Block Web Programs From Being Viewed on Google TV. http://online.wsj.com/article/SB10001424052702303339504575566572021412854.html, 2010-10-22, retrieved 2010-11-02.

[8] Showtime ITV. Showtime interactive tv. http://www.sho.com/site/itv.do, retrieved 2010-11-12.

[9] Ben Drawbaugh. An inside look at a verizon fios super headend and video hub. http://hd.engadget.com/2009/12/17/an-inside-look-at-a-verizon-fios-super-headend-and-video-hub/, 2009, retrieved 2010-11-12.

[10] BBC Trust. Bbc red button service review. http://www.bbc.co.uk/bbctrust/assets/files/pdf/regulatory_framework/service_licences/service_reviews/red_button/red_button_final.pdf, 2010.

[11] VGChartz. Hardware totals. http://www.vgchartz.com/hardware_totals.php, retrieved 2010-11-30.

[12] Yinka Adegoke. Microsoft eyes leap back into tv: sources. http://www.reuters.com/article/idUSTRE6AS2E120101129, 2010, retrieved 2010-11-30.

[13] AT&T. AT&T Extends TV Watching to More Devices with Launch of U-verse TV on Xbox 360. http://www.att.com/gen/press-room?pid=18642&cdvn=news&newsarticleid=31276, 2010, retrieved 2010-11-30.

[14] The Nielsen Company. Game consoles edge closer to serving as entertainment hubs. http://blog.nielsen.com/nielsenwire/online_mobile/game-consoles-edge-closer-to-serving-as-entertainment-hubs/, 2010, retrieved 2010-12-17.

[15] W3C. Xml essentials. http://www.w3.org/standards/xml/core, retrieved 2010-12-14.

[16] Cisco. Cisco visual networking index: Forecast and methodology, 2009-2014. http://www.cisco.com/en/US/netsol/ns827/networking_solutions_white_papers_list.html, 2010.

[17] Phil Turner David Benyon and Susan Turner. *Designing Interactive Systems*. Addison-Wesley, 2005.

[18] Douglas Crockford. Private members in javascript. http://javascript.crockford.com/private.html, 2001, retrieved 2011-02-03.

[19] Chen Y.-P. P. Tjondronegoro, D. and A. Joly. A scalable and extensible segment-event-object-based sports video retrieval system. 2008.

[20] ITU-T. ITU-T Recommendation H.740, Application Handling for IPTV services. http://www.itu.int/rec/T-REC-H.740-201003-I/en, 2010, retrieved 2010-11-09.

[21] Cisco. The evolving internet. newsroom.cisco.com/dlls/2010/ekits/Evolving_Internet_GBN_Cisco_2010_Aug_rev2.pdf, retrieved 2010-12-15.