# PriSEMD - A Privacy-Friendly Approach to Analyze and Measure Smart Entertainment Devices

**PriSEMD - Ein datenschutzfreundlicher Ansatz zum Analysieren und Messen von Smart Entertainment Devices**

Master-Thesis von Jan Müller aus Darmstadt

Dezember 2014

TECHNISCHE UNIVERSITÄT DARMSTADT

Fachbereich Informatik
Fachgebiet Sicherheit in der Informationstechnik

CASED

EC SPRIDE

PriSEMD - A Privacy-Friendly Approach to Analyze and Measure Smart Entertainment Devices
PriSEMD - Ein datenschutzfreundlicher Ansatz zum Analysieren und Messen von Smart Entertainment Devices

Vorgelegte Master-Thesis von Jan Müller aus Darmstadt

Prüfer: Prof. Dr. Michael Waidner
Betreuer: Marco Ghiglieri

Tag der Einreichung:

# Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den 14. Dezember 2014

(J. Müller)

## Zusammenfassung

Smart Entertainment Geräte(SED) gehören zu dem Bereich der Unterhaltungselektronik und können heutzutage in vielen Haushalten vorgefunden werden. Von herkömmlichen Geräten der Unterhaltungselektronik unterscheiden sich SEDs vor allem darin, dass sie Schnittstellen zur Verbindung mit dem Heimnetzwerk/Internet zu Verfügung stellen. Hinzu kommen zahlreiche neue internetbasierende Funktionalitäten, wie zum Beispiel das direkte Abrufen von Fernsehsendungen aus Mediatheken. Auf der einen Seite bieten diese Funktionen Vorteile, sie könnten aber auch beispielsweise dazu genutzt werden, um das Nutzungsverhalten von Konsumenten detailliert aufzuzeichnen. Hersteller und die Werbebranche sind an diesen Daten interessiert, um zielgerichtete Verbesserungen an ihren Produkten bzw. Werbung durchführen zu können.

In dieser Arbeit stellen wir ein Konzept vor, welches die gesendeten und empfangenen Daten von SEDs sammelt und diese nach einem festen Regelwerk auswertet. Die Implementierung eines Prototyps zeigt die praktische Realisierbarkeit dieses Konzepts an Hand der beispielhafen Analyse eines Smart TVs. Zu Evaluierungszwecken wurde der Prototyp in einem realen Haushalt getestet. Die Ergebnisse dieses siebentägigen Tests belegen, dass es möglich ist Daten von SEDs zu sammeln und Aussagen über das Konsumentenverhalten zu treffen, wie z.B. welche TV Sendung vom Konsument bevorzugt angeschaut wurde.

Ein weiterer Teil der Arbeit befasst sich mit einem Konzept, bei welchem Daten an Dritte weitergeleitet werden können ohne dabei die Privatsphäre des Konsumenten zu verletzen. Der Konsument hat die Möglichkeit die Weitergabe der Daten zu kontrollieren, indem er die Weitergabe für bestimmte Drittparteien authorisiert oder diese generell untersagt.

## Abstract

Smart Entertainment Devices (SED) are belonging to Consumer Electronics (CE) and can be found in many households nowadays. SEDs can be differentiated from CE devices with several aspects. The most important difference is that SEDs provide interfaces to connect to the local network/Internet. Internet-based functionalities like accessing TV programs in media libraries are new features of SEDs. These new features represent on the one hand benefits for the user, but on the other hand, it is possible to track user behavior. Device manufacturers and advertising companies are interested in this data for e.g. improving their products or adverts.

In this work we present a concept which allows gathering data sent and received by SEDs. In a succeeding phase, the gathered data is evaluated based on a defined set of rules. The practical and technical feasibility of this concept is shown by a prototype implementation.

Due to evaluation purposes, the prototype was tested in a realistic household. This test includes an exemplary analysis of a Smart TV. The results of this 7-day-test have proven that it is possible to measure data from SEDs and to find characteristics concerning user behavior. Characteristics that can be stated are for example the favorite TV programs of a user.

In another part of this work, a concept is presented which shows how to forward PriSEMD measured data to third parties without violating the user's privacy. The SED user has the opportunity to control the transferring process. It is possible to approve certain third parties or to prohibit data transferring at all.

## Contents

# 1 Introduction

The Smart Home paradigm is a topic for in house control and represents an all in all attempt to integrate energy supply, white ware, computer and entertainment devices. A lot of companies offer products in this area. Samsung, for example, sells a smart washing machine with integrated WLAN functionality for controlling the washing process remotely [6]. Another example is LG which offers a kitchen solution consisting of a smart oven and refrigerator which can be controlled by a smartphone [7]. Due to a statement of the Allied Market Research Organization, the volume of the Smart Home market will be growing around 30 percent annually until 2022 [4].

In this work the focus is on a special group of smart devices, the Smart Entertainment Devices (SED). SEDs belong to the class of Consumer Electronics (CE). Representatives of SEDs are for example DVD/Blu-Ray players, set-top boxes, video gaming consoles or Smart TVs (STVs). SEDs are differentiated from conventional CE devices, because they have more interfaces available and more functionalities in regard to user device interaction. One of the most influencing differences is the integration of a network interface allowing communication with other devices and data exchange with parties located in the Internet.

The integration of network interfaces in SEDs is leading to new interactive functionalities. Music and movies can e.g. be directly accessed and played back by SEDs. The SED user has the opportunity to request additional information from the Internet and display them on the SED. The Hybrid Broadcast Broadband TV (HbbTV) standard extends for example STVs with new opportunities to receive additional program information [35].

However there are also disadvantages concerning these new technologies. The increased network capabilities can be misused to send data from SED to interested stakeholders. One example is the information retrieval of LG STVs published in a news article of November 2013 [39]. The affected STV models were gathering data concerning viewing behavior and names of files saved on external storage devices. This data was sent to the manufacturer using an unencrypted data connection. Due to statements from LG, the harvested data was used for advertising and TV program recommendations. An update released one month later introduced an encrypted data connection and prevented the STVs from gathering file names, but nevertheless, user viewing behavior is still recorded and transmitted by the STVs [40].

Another example of recording and transmitting user behavior data shows a security test performed by Ghiglieri, Oswald and Tews [25]. Mainly responsible for that data transmission is the previous stated HbbTV technology. The data is sent to the STV's manufacturer, TV stations and Google Analytics. One important fact is that the data transmission is executed by default. The only requirement needed is a working Internet connection and a HbbTV capable device.

Both cases show that user behavior related data is produced and transmitted without consent of the user. In addition, these processes run per default and options to disable these processes are missing. One objective of this thesis is to inform the SED user which data is produced and transferred by SEDs. Another objective is to provide the user a possibility to make a decision which data is transmitted and which not.

## 1.1 Contributions and Thesis Structure

The Privacy friendly approach to Analyze and Measure Smart Entertainment Devices (PriSEMD) is introduced in this master thesis.

PriSEMD is a new concept of audience measurement respecting the privacy of SED users. For parties analyzing the audience measurement results it is not possible to determine the exact origin of the analyzed user. In contrast to other audience measurement systems, PriSEMD supports current technologies, e.g. the analysis of Internet traffic.

The PriSEMD concept describes how to measure data from SEDs in order to obtain information about the user behavior. It is explained how data can be measured, processed, analyzed and presented. The implementation of a prototype and a performed evaluation test proves that the theoretically explained PriSEMD concept can be applied in a realistic scenario.

The measurement results of PriSEMD are used for two purposes. First, results can be inspected by SED users in order to get an insight of data produced by SEDs. Many SEDs are transmitting data under the surface, usually without knowledge and consent of the user. PriSEMD provides an opportunity to inspect the data produced and transmitted within these background processes. Second, the results of PriSEMD can be shared with third parties, e.g. device manufacturer or service providers. However, the sharing process can be controlled by the SED user.

The main questions answered in this Master's thesis are the following:

- Which data produced by SEDs can be measured?
- Which of the data can be used to predict user behavior?
- Which facts about preferences, interests and habits of the user can be stated while analyzing that data?
- Is it possible to find any tradeoff between companies which are interested in using the measured data and the privacy of SED users?
- How can such an approach be implemented?

The structure of this thesis is divided into a theoretical and an implementation part. The concept of PriSEMD is generally outlined in the theoretical part. The implementation part focuses on a prototype which implements the main aspects of the PriSEMD concept.

After an introduction and a definition of the major questions answered within this thesis, a detailed description of SEDs follows in Chapter (2). The description consists of a general definition of the term *smart* and a distinction between smart and non-smart devices. Afterwards, the usage differences between SEDs and non-smart entertainment devices are discussed (2.1). In the end of the chapter, a Smart TV is introduced as an example for a SED (2.2).

In Chapter (3) the concept of PriSEMD is outlined. The detailed structure of PriSEMD is explained in (3.1) and consists of a detailed description how to measure, analyze and present SED data. Three interfaces of SEDs are introduced in Section (3.2) focusing on requestable and measurable data. The privacy aspects relating to PriSEMD are discussed in Section (3.3). It includes a comparison of SED user and third party interest, a depiction of attack vectors to the SED users privacy and a concept for anonymous data transmission.

The implemented prototype is presented in Chapter (4). In the first part of Chapter 4.1 the network architecture is explained where the prototype is implemented in. Furthermore, a short description of the required hardware components is stated. The detailed software implementation of PriSEMD is presented in (4.2) where each software component is explained in detail and illustrated using examples.

In several parts of Chapter (3) and (4), text boxes provide necessary background information.

Chapter (5) is divided into two parts. The first part 5.1 is an evaluation of an one week test during the prototype was deployed in a test environment. The second part 5.2 introduces four more SEDs and examines which information can be measured from SEDs in general. Therefore, the focus is not set to a specific device but rather to all SEDs in general.

At the end of this thesis the conclusion, future work (6) and appendix (7) are following.

## 1.2 Related Work

To the best of our knowledge this thesis is the first attempt of measuring Smart Entertainment Devices while respecting the privacy of the user. The related work section consists of already published work in the areas of audience measurement and privacy in SEDs.

Audience measurement is a technique to measure user behavior data from e.g. television devices. The Nielsen company and the „Gesellschaft für Konsumforschung“ (GFK) are two companies specialized in TV audience measuring.

The Nielsen company[1], e.g. working for the US market, uses so called *people meter* devices to measure TV viewing behavior. Those devices are installed in families selected from a random sample. Each *people meter* is a box that is connected to the TV in order to measure the currently running channel. Every person has to press a button on the device while watching TV. This allows to distinguish individual persons who are watching TV. The measured data is transferred to Nielsen servers using the telephone connection of the family's household [49, p. 333].

The GFK[2] is running TV audience measurement in Germany. For the measurement purpose so called *GFK-Meter* devices are installed in 5640 German households . The *GFK-Meter* is a device connected with the TV signal. It is able to meter the current running TV Channel, an optionally used VCR, either receiving or playing a program, and the usage of video text. Individuals can be distinguished by pressing a personalized button provided on a remote control. The data measured by the *GFK-Meter* is transmitted at night via telephone connection to GFK servers [5, pp. 179-180].

The attempt discussed in this thesis tries to focus a greater context involving more devices from the area of Consumer Electronics. New technologies as for example Internet are considered in the concept of this thesis. Both concepts mentioned above are focused on measuring TV signals only. Moreover, they are not respecting new technologies coming along with the Internet connection of TV devices. From the privacy point of view, the data transmission via telephone connection can be seen controversially. The telephone connection is related to a unique telephone number which can be used to identify a household. This makes it possible to match the measured data to a specific household.

An attempt related to several aspects of the concept introduced was shown by Drosatos, Tasidou and Efraimidis [13]. Drosatos et al. outline a concept called *Privacy-Preserving Television Audience Measurement* (PrivTAM). All in all, PrivTAM is a TV audience measurement system with respecting privacy of the measured individuals. The data transmission, techniques to anonymize data and introducing an appropriate system architecture are focused within their paper. Due to restrictions of the used STV (has to be based on Android), the attempt from Drosatos et al. is more specifically than the general concept discussed in this thesis.

Concerning multimedia device measurement a few patents can be found, e.g. [41, 29]. Both concepts are older than 10 years and do not consider recent challenges, e.g. measuring computer network communication.

The HbbTV technology is used for requesting additional program information on STVs. This technology can further being used for recording the STV's user behavior [24, 26, 25]. The main problem found in those works is that the data transferred to other parties cannot be controlled by the user. The user for example does not have the possibility to control

[1] http://www.nielsen.com

[2] http://www.gfk.com

the data to the TV stations. Deactivating the functionality or disconnecting the STV from the Internet reduces the range of functions and is often not intended by the user. PriSEMD offers SED users to control and inspect data produced by SEDs.

The Universal Plug and Play (UPnP) protocol is commonly used by SEDs. UPnP allows to communicate easily with SEDs and access their provided services. It should be noted that a few vulnerabilities are found in the past [19, 34]. The flaws of UPnP are not considered in this thesis but data produced by UPnP services is being processed.

## 2 Smart Entertainment Devices

A lot of network connected embedded devices are invading our everyday life [10]. These devices are often called „smart“ devices. An example for the transition is the change from conventional televisions to Smart TVs. The trend goes in the direction that more and smarter devices are emerging [27]. Before explaining what Smart Entertainment Devices are, it is important to clarify what is meant by the term „smart“. After stating a possible definition of „smart“, features and characteristics of smart devices are outlined. In addition, to the presentation of some examples, a distinction from other devices is made. A definition of the prefix smart could be read as follows:

> *The term „smart“ describes something as intelligent and able to autonomously acquire and apply knowledge [8, p. 3].*

Explained with the example of the conventional cellular to smart phone transition, a smart phone has more features, allowing the user to interact with the device. The implementation of Internet and the availability of apps are two functionalities differing cellular from smart phones. In the early 90s, no cellular phone was able to access to the Internet or having apps. Today, these two features are the most significant additional features, accompanied by the transition to smart phones. In the next paragraph it is shown which features and aspects are characterizing smart devices.

A traditional device differs from a smart device in many ways. More computing power and more interfaces to interact with the environment are the two important differences. The enhancement of the computing power leads to different changes. Now it is possible to run modern operation systems. Smart mobile devices are often based on Android, iOS or Windows, other smart devices like Samsung Smart TVs are using a Linux based operating systems [42]. Accompanied by this fact, a lot of new features are available. These features, previously only known from the computer area, are now integrated in smart devices and open a new dimension of opportunities.

The availability of interfaces to interact with the environment is closely associated with the increasing computing power. The processor is now able to handle different tasks. A huge impact is the availability of network interfaces. Either cable bound or wireless allow the smart device to communicate with other devices. On the one hand, there are communication partner in the Local Area Network (LAN), on the other hand, devices in the Internet can be requested.

After describing the meaning of the term smart in general, the focus of this thesis is set to a special group of smart devices, the Smart Entertainment Devices (SEDs). There is no exact segregation of what is called a SED. In the context of this work, SEDs are Consumer Electronic (CE) devices. Figure 1 shows a few examples for SEDs. Besides Smart TVs, which will play a major role in this thesis implementation part, there are still other devices in the group of SEDs. Set-top boxes, stereo sound systems and video gaming consoles are e.g. SEDs. The collection of the shown examples is not exhaustive. However, there are devices that do not belong to the group of SEDs and are therefore not focused in this thesis. Computer devices, e.g. notebooks and desktop computers, as well as smartphones and tablet computers, are explicitly no SEDs.

The influence of SED devices in comparison with the people using them is described in Section 2.1. The Smart TV as an example for a SED is focused in Section 2.2.

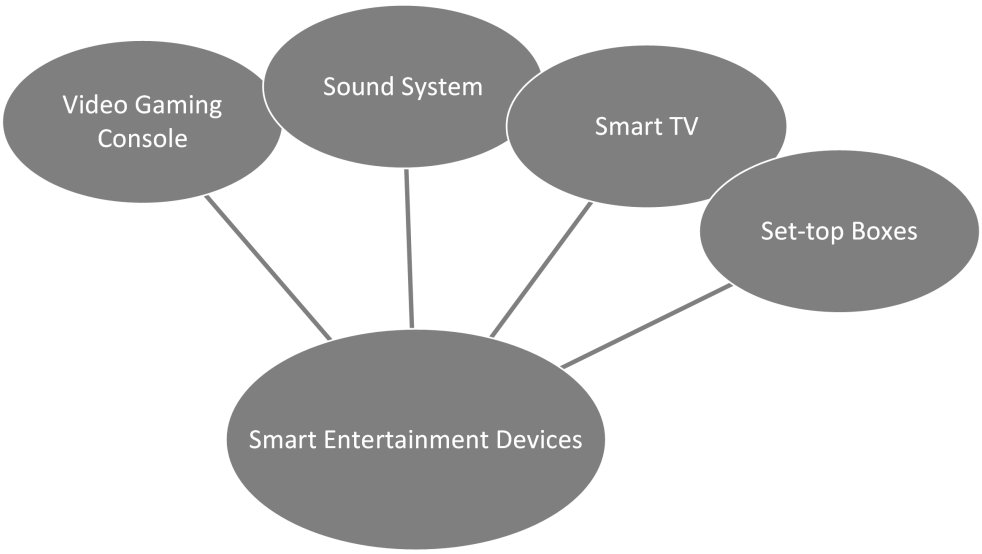



**Figure 1:** Smart Entertainment Device Examples

## 2.1 SED and Consumer

At the first sight, there are a lot of new opportunities going along with the smart revolution. The connection of SEDs offers the possibility to access own content from almost every SED. Own data, like music and video files, can be shared between SEDs. Movies stored in a Network Attached Storage (NAS), can be for example used from Smart TVs, computers or from smart watches. Another new opportunity is controlling SEDs. Today, an infrared remote control is not the only available option to send interaction commands to the SED. It is possible to use devices like tablets or other devices to control your favorite SED. The Android App SamyGO sends e.g. control messages via network to a STV [1].

Besides the numerous advantages described above, there are still doubts with this kind of new technologies. One of the repeatedly occurring statements is the fear of privacy violation. Due to a study of „Gesellschaft für und Unterhaltungs- und Kommunikationselektronik (GFU)“, the privacy aspect leads 26% of all Germans to disconnect their Smart TV from the Internet [32].

Furthermore, users are unconfident if they are going into an addiction of their devices. More and more technical devices are used in private environments, as for example in the living room of the user's home. Besides the mentioned Smart TV, there are smart disc players, smart video gaming consoles and smart set-top boxes like Apple TV.

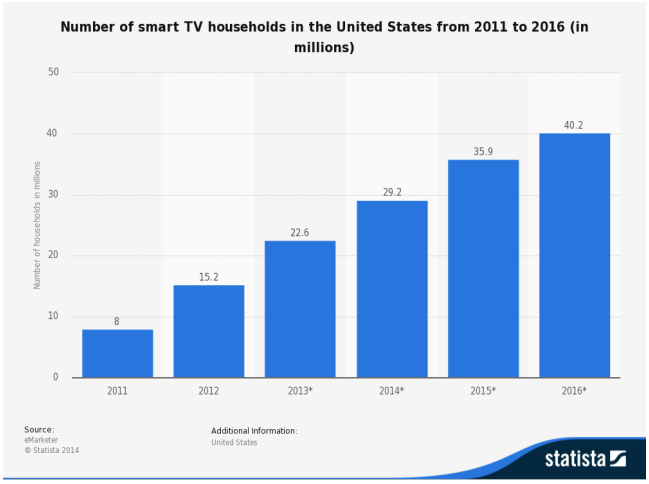
Another aspect stated in the study of GFU [32] is a comparison of conventional and STVs usage behavior which shows that it is changed in several ways, e.g. users consume fewer programs, broadcasted at a specific time - they use more on demand services in order to watch their favorite program.

SEDs are used to consume entertainment and therefore usually running in the leisure time. While using SEDs, personal preferences of the user are always focused such as watching favorite TV programs or listening to preferred music. Other devices like a smart oven or washing machine are used for one particular use case; cooking or washing. This leads to the fact that SEDs are closer to the private life than other computer devices. Due to the fact that SEDs are closer to users' private lives, SEDs are able to gather more data about habits and preferences than other smart devices applied in the user's home.

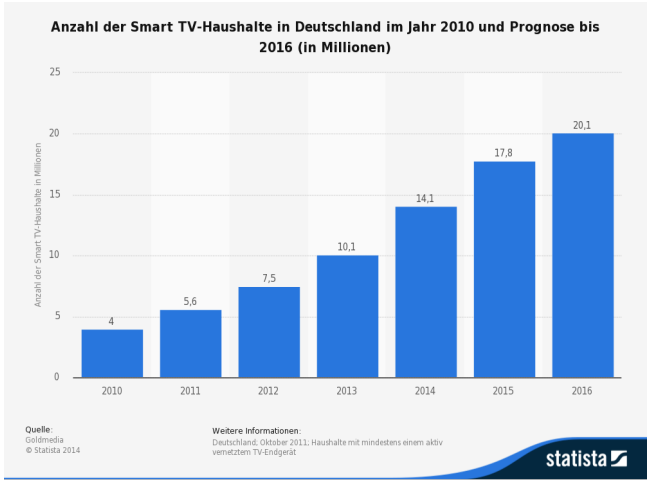
## 2.2 A SED Example - Smart TV

Smart TVs (STV) are one of the first widely accepted SEDs. The statistic shown in Figure 2a and 2b shows that the dissemination of STVs will continue to grow. Figure 2a shows the number of US households owning a Smart TV related to the year. From 2011 to 2016 a rise of STV using households is recognizable. The numbers of STVs households is growing by 27.37 per cent from 2014 to 2016. Figure 2b shows a similar situation in Germany. Here, the number of STV using households are growing by 29.85 per cent from 2014 to 2016.

All in all, the Smart TV market is growing and will gain more influence in the future.




**(a)** Smart TV Statistic - USA [45]




**(b)** Smart TV Statistic -Germany [46]

As an example for SEDs the Smart TV is described in the remainder of this section. The first part is about describing the variety of new features. Following, user and Smart TV relation, benefits and disadvantages using Smart TVs and the transition from traditional TV to Smart TV are outlined. In the end, a brief insight into the privacy aspects is given.

Compared to conventional television devices, a STV is an improved version. Watching TV is not anymore the unrestricted main feature of a STV. It is more a functionality besides a lot of others forming the new picture of a television device. The listing below shows several new features:

- **Additional program information**: The user is able to receive additional program information while using a Smart TV. Conventional devices have the teletext; STVs have the opportunity to receive a huge amount of additional in-

formation via the Internet. One way to establish this functionality is the standardized Hybrid Broadcast Broadband TV (HbbTV) explained in Chapter 3.2.1. This functionality is independent from the manufacturers and supported in almost every new STV.

- **Software/apps**: Accompanied with the increasing computing power, it is possible to run a variety of different software on the Smart TV. Often these programs are called apps, short for applications. In the context of SEDs, apps are usually manufacturer specific and can be used for example to access media libraries. Nevertheless, there are a lot of other possibilities which users can do with software on STV. It is possible to compare the STV software with apps for mobile devices. Software for news, services like Skype or Facebook and weather forecast are e.g. common use cases.
- **Video on demand**: It describes the opportunity to access television content when the user wants to, not when it is broadcasted. On the one hand, there is a way to access media via a media library like „ZDF - Mediathek“ or „ARD - Mediathek“. This additional content is usually provided via Internet and can be accessed by using apps. Furthermore, it is often possible to access the media library from other devices like computers or mobile devices. This blurs the boundary between Smart TV and other devices. On the other hand, television content and movies can be accessed by online video stores. A representative of this is for example Netflix or Amazon Instant Video.
- ...

The accessibility of Internet leads to the fact that many known Internet functionalities are available for STVs. Sending and receiving emails, chatting and using social media networks can now be done with STVs.

Figure 2 shows what interfaces are available on today's Smart TVs. All interfaces can be divided into two groups. The group of unidirectional and bidirectional interfaces. The unidirectional interfaces involve the interfaces for receiving the TV signal. Depending on which signal method is used, either cable bound, satellite or terrestrial an appropriate standard is used by the STV. Besides the analogue TV, which is no object of this work, there are the Digital Video Broadcasting (DVB) standards DVB-C, DVB-S and DVB-T. Another representative of the unidirectional interface group is the infrared interface. This interface allows a user to send control messages to the TV while using an appropriate infrared remote control.

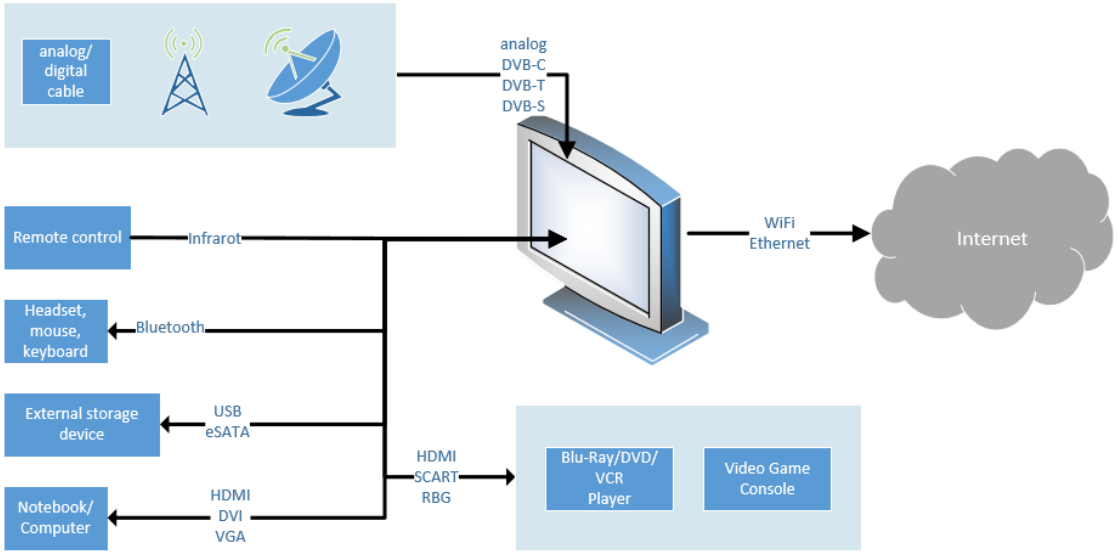



**Figure 2:** Smart TV Interfaces

Nevertheless, the group of bidirectional communication methods is larger. There are interfaces for data exchange like USB, eSATA and interfaces to share audio/video signals like HDMI, DVI and VGA. Furthermore, a lot of Smart TV devices support Bluetooth. With Bluetooth it is possible to connect 3D glasses or a headset to the STV. However, the most important interface is the network interface. The network interface can be either wireless (Wi-Fi) or cable bound (Ethernet). Using this interface, the STV has the opportunity to communicate with other devices. A lot of software features, stated above, are using the network interface to receive information about the Internet. It is as well possible to send messages to devices located in the Internet.

Besides the Internet functionality, the Smart TV is able to communicate with other devices in the LAN. A STV is for example able to communicate with a tablet or smartphone in order to receive remote control signals of them. Furthermore, video gaming consoles can be paired with the STV to play games on the STVs display.

## 3 Concept of PriSEMD

PriSEMD is a concept describing how to collect, evaluate, analyze and present data of SEDs. The concept is not limited to a specific device. It can be applied to any device belonging to the group of SEDs. PriSEMD covers the analysis of *n* different SEDs which have to be connected to a home network and further to the Internet. The PriSEMD concept could be deployed for example in homes or offices. In offices, TVs might be used for presentations or displaying information - often they replace data projectors.

One of the key aspects of the PriSEMD concept is to collect free accessible data from SEDs. In the context of PriSEMD, free accessible means that no attacks or attack like methods are being used in order to measure data from SEDs. All collected data can be measured either passive or active. Passive measurement includes for example the collection of network data. It is called passive, because PriSEMD does not actively communicate with the measured SEDs. In the case of network communication PriSEMD only listens to the network traffic of the measured SEDs. Active measurement includes an interaction between PriSEMD and the measured SED. One possible active measurement process could be requesting data of a SED while sending a request message to a UPnP service and recording the response message of the SED. Further details about data measurement of UPnP are stated in Chapter 3.2.2.

Another aspect of PriSEMD is the analyzing process of the measured data. A major goal of PriSEMD consists of extracting and processing user behavior related information in order to make it accessible for interested parties.

All in all, there are two different categories of stakeholders concerning PriSEMD. The first category consists of the **local user(s)**. The local user is the owner of the measured SEDs and having installed PriSEMD for example in his/her home. One possible motivation of installing PriSEMD at home could be monitoring SEDs. Data produced by SEDs and transmitted to other devices either in the local network or Internet are collected and analyzed. Finally, the results are presented to the local user via a user interface. The presentation follows the purpose to inform the local user about which data is produced by the analyzed SEDs. This achieves a degree of transparency, because invisibly collected data is made visible to the user. Even a local user who is not familiar with network/Internet technologies can benefit of this presentation.

**Third party users** belong to the second group of stakeholders. Usually, they do not have access to the local network and are potentially not trustworthy. SED manufacturers and service providers are for example third party users. Both are interested in metering the user behavior in order to improve their products and services. A discussion of the stakeholders' user interests can be found in Chapter 3.3.1

In the further description it is important to differentiate between the terms data and information. Before describing the approach of PriSEMD in detail, a short definition of those terms is given:

- „**Data** is a collection of raw, unorganized facts that need to be processed. Data can be something simple and seemingly random and useless until it is organized.“ ([56])
- „When data is processed, organized, structured or presented in a given context to make it useful, it is called **information**.“ ([56])

In the context of this thesis, unprocessed data measured from the SEDs is called data. The results of the processing are called information, e.g. the user behavior related data.

In Section 3.1 the PriSEMD concept is explained in more detail. Therefore, the concept is divided into a step by step description of collection, evaluation, analysis and presentation phases. Within Section 3.2, three typical SED interfaces are introduced focusing which data can be collected while measuring. Furthermore, it will be outlined which information can be extracted out of the measured data concerning user behavior. The privacy aspects concerning PriSEMD are discussed in the last Section 3.3. Besides stating attack vectors affecting the local user's privacy, an approach to anonymously transmit data to third party users is introduced.

### 3.1 Concept Details

The idea of the metering process is explained within this chapter. First, an overview of the PriSEMD network architecture is given. Finally, the PriSEMD entity is explained in more detail.

Figure 3 displays the network architecture required by the PriSEMD concept. The SED devices are on the left side. As mentioned above *n* different SEDs are supported within PriSEMD. All SEDs are connected with the PriSEMD entity in the middle of the figure. The PriSEMD entity is further connected to the Internet. The reason for this setup is to establish PriSEMD as a proxy device, i.e. any traffic coming from a SED must pass the PriSEMD device. This allows PriSEMD to access and read the network traffic without interrupting the connection between SED and Internet.

As shown in Figure 4, the software concept of PriSEMD can be divided into four different phases. The Collection Phase is the first part being executed. Here, the necessary raw data is measured from the SEDs. The Processing Phase is executed afterwards. Any raw data collected from the phase before is processed in this phase and converted into a consistent structure. The Analyzing Phase uses the created structure before and extracts user related information. The Presentation Phase is the last phase in the cycle where a user interface displays all results.

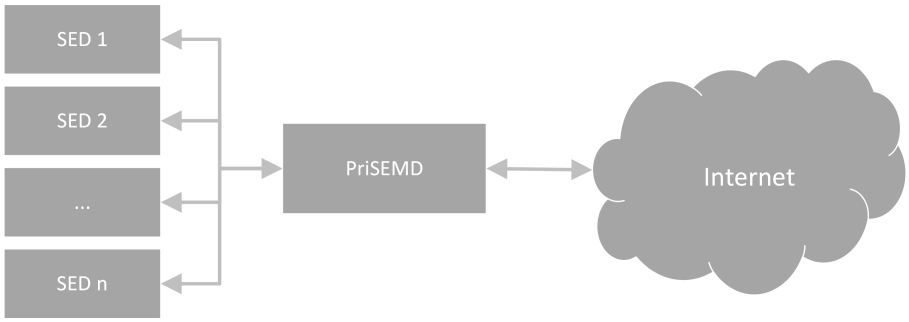



**Figure 3:** Conceptual Network Architecture


**Figure 4:** Concept Overview

A more closely look at each phase will be provided in the following four subsections. The detailed concept shown in Figure 5 is used in the these subsections to describe the functionality of PriSEMD in detail.

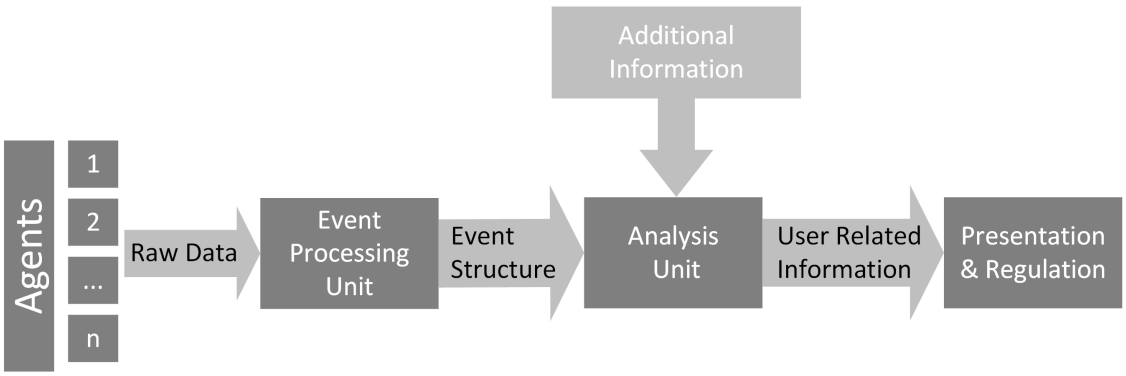



**Figure 5:** Detailed Concept

### 3.1.1 Collection Phase

Data measurement of SED devices is done in the Collection Phase. A SED is measurable if it has at least one interface which produces data. PriSEMD is able to measure the interface, if the prerequisites are fulfilled. Measuring network communication requires PriSEMD for example to have a network interface and be able to access the local network. Another example is eavesdropping infrared signals. PriSEMD is able to measure these signals if it has an infrared sensor implemented. A detailed description that can be measured in context of SEDs is outlined in Chapter 3.2.

**Identifying a SED**

Before an agent is able to measure data of a SED, a unique ID has to be established. This ID consists of the MAC and IP address of the SED and the process of identifying requires that the SED is turned on. After checking the active device status, the ID is assigned to the SED. It should be noted that the MAC and IP address of the SED device can be changed or manipulated. In order to cope with a changed IP address, the ID has to be renewed after a period of time. Therefore, in the remainder of this work, we assume that no manipulation of MAC has been made.

The activity status of a SED can be discovered while analyzing network traffic. As described later on, network messages based on the SSDP protocol can be e.g. analyzed to check the activity status of a SED. More details on the concept of identifying SEDs are outlined in Chapter 3.2.1.

**Agents**

The measurement process is performed by programs called **agents**. Each agent is responsible for measuring one specific interface. Additionally, an agent is able to measure a specific interface of multiple SEDs, but only if the interfaces are the same.

One agent is needed e.g. for measuring network traffic and one for infrared signals. Both agents are able to consider multiple SEDs. The network traffic measuring agent is able to consider any SED which is producing network traffic. The same applies to the infrared agent which analyzes any SED using infrared signals.

An agent is autonomously running and not affected by other agents. As shown in Figure 5, PriSEMD supports *n* different agents. One agent has to be implemented at least within PriSEMD, otherwise there is no data that can be processed by the next phases. Each agent is permanent running in order to measure data twenty-four-seven. When an agent is halted, it is no longer able to measure data. Due to the fact that the data measurement shall be permanent, the agent software has to run stable.

Agents can be distinguished by their measurement techniques, either **active** or **passive**:

- A **passive agent** measures data in real time and without requesting it from a SED. Data measurement is limited to data sent or received by SEDs. A passive agent measures data only when it is available, meaning if a SED does not send or receive data the agent is not able to measure anything. The network analyzer agent is for example only able to measure network traffic if the SED sends or receives network messages.

- An **active agent** measures data while requesting data from the SED. The agent sends a request message to the SED and analyzes the responded message which contains the desired data. In contrast to a passive agent, an active agent is triggered in a specific time interval. The time interval can be chosen between milliseconds and minutes and up to hours. The intervals must be adjusted based on different factors. First, it is important to avoid high network traffic for the reason to not negatively affect the network communication of others. Second, the interval should be selected so that it does not interfere with the SED. The SED has to be able to process the requests and shall not stick in an overload. It is important to avoid high network traffic and to overload the SED with requesting messages, because this negatively affects the performance of the SED services.

#### Raw Data

The term **raw data** is a topic for any data collected by the agents. It does not describe the network traffic recorded. A **raw data set** is the smallest entity of raw data. The collection of any raw data set represents the raw data of an agent. Each raw data set consists of three different parts described in the listing below:

1. **SED Device Name**: This part represents the name of the source SED device. The SED Device Name consists of a general SED description containing manufacturer name and the model number. The STV used in this thesis has the following SED Device Name: „STV Samsung UE40ES6300“.

2. **Timestamp of occurrence**: This part represents the date and time when the raw data set was exactly measured. The timestamp consists of information about year, month, day, hour, minute and second and is formatted as follows: „YYYY-MM-DD HH:MM:SS“. An example timestamp could look like „2014-07-18 11:21:59“. The time zone GMT+1 is used in this work.

3. **Measured information**: This part of the raw data set is the payload information. Depending on the purpose of the agent, this section contains different sort of data. In context of network analysis, details about e.g. used protocol could be e.g. stated here.

#### Measuring Procedure

The performed measurement procedure of an agent can be described as follows. Depending on whether being active or passive, the agent runs through another process. Figure 6 outlines the measurement procedure of an active and of a passive agent. Four different entities are outlined in Figure 6: a SED, a passive/active agent and a database.

The listing below describes the procedure displayed in Figure 6. The active and passive agents' procedures differ in two points. The active agent executes the first step whereas the passive agent skips it. Further, the active agent is triggered in a specific interval and the passive agent is running in real time.

1. **Request Data**: An active agent sends request messages to a SED interface. The request message is depending on the measured interface and the type of requested data.

2. **Receive Data**: An active agent waits for response messages, whereas a passive agent measures messages that are sent from or to the SED. After receiving a message, a raw data set is created using the data contained in the message.

3. **Processing**: This step examines if the raw data set object contains correct data. If this test ends with a valid result, the data set is forwarded to the database (step 4). Otherwise, the raw data set will be dropped. The test fails if e.g. certain data is not included in a response message of a requested SED service.

4. **Storing**: The processed data from the previous step is saved in the database.

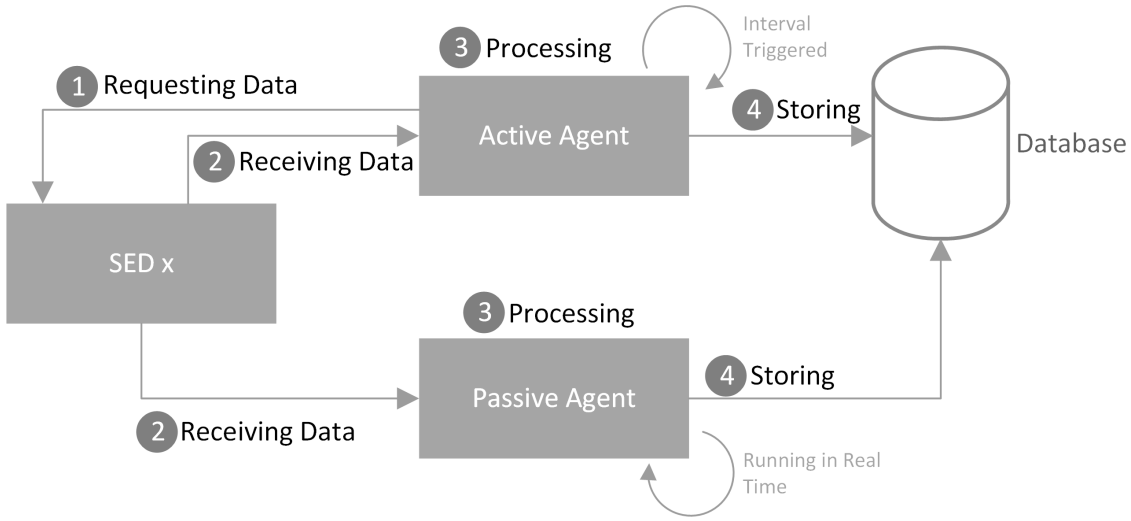



**Figure 6:** Agent Procedure

### 3.1.2 Processing Phase

The Processing Phase is the second phase within the PriSEMD concept. The goal of this phase is to merge the raw data collected by the agents of the Collection Phase into a consistent structure, called event structure. For this purpose, the raw data is converted into events. The strategy how to process raw data and create events is described by rules. Figure 7 provides an overview about the procedure of this phase. The **Event Processing Unit (EPU)** is the central component here. The EPU receives the raw data gathered within the Collection Phase and calculates events while using provided rules. Events, rules and the EPU converting procedure are described in the following paragraphs.

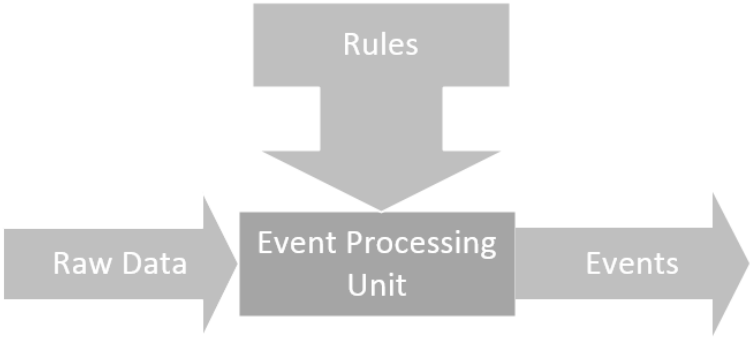



**Figure 7:** Event Processing

**Events**

An **event** is an action extracted from a recorded raw data set. Each event consists of three parts described in the following listing:

1. **Event Description**: The event description is a text describing the action of an event. *SED x turned on* is an example event description text used in context of the later discussed PriSEMD prototype.

2. **Event Timestamp**: Similar to the raw data set explained in Chapter 3.1.1, each event has a timestamp describing when the event happened. The structure of the timestamp is „yyyy-mm-dd HH:MM:SS“.

3. **Event Level**: The level of the event structure is stated in this field, e.g. 1, 2, 3.

The events are organized in a level based hierarchy called **event structure**. Starting with level 1, it is possible to define events up to level $n$. Events lower than level 2 are connected to a higher level event. Level 1 event do not have an

upper level event layer, because it represents the highest possible level. In other words, an event level *n-1* has to have an event level *n* to which it is connected to. A three-level event structure is used for further descriptions. Level one event consists of SED devices, level two of major events occurring in context of a SED and level three of minor events occurring in context of a major event.

| Level 1 | Level 2 | Level 3 |
|---|---|---|
| SED *x* | major event 1<br>major event 2<br>major event 3 | minor event 1 |
| SED *y* | major event 1<br>major event 2 | major event 1<br>minor event 2<br>major event 1 |
| SED *z* | SED turned on<br>SED turned off | Infrared key: POWER pressed<br>HTTP Messages detected |

**Table 1:** Event Structure Example

Table 1 shows an example event hierarchy. There are shown three different SEDs, SED *x*, *y* and *z*. As described above, the SEDs are representing the events level 1. SED *x* has four sub events consisting of three major events in level 2 and one minor event in level 3. All major events are connected with SED *x*. This means, the events are occurring in the context of SED *x*. The minor event 1 is further connected to major event 2. The event structure for SED *z* displays a few example events. Two possible major events are *SED turned on* and *SED turned off*. The two minor events *Infrared key: POWER pressed* and *HTTP message detected* are connected with the major event *SED turned on*.

#### Processing Procedure and Rules

The processing procedure is executed by the EPU and will be introduced in combination with a rule description in this paragraph.

Algorithm 8 displays the high level procedure of the EPU. As already shown in Figure 7, the EPU obtains raw data sets and rules as input and calculates events as output. The rule consists of instructions how to build events and which raw data sets are required. Algorithm 8 iterates over each provided rule and calculates events as specified in the rules. After selecting one rule, all relevant raw data sets are searched. In the next step, an appropriate event from the upper level is searched for each raw data set. If an upper level event was found, the event is constructed and saved.

```
input  : raw data set RDS, rule R
output: event E

for each provided R do
    read all RDS stated in R;
    for each RDS do
        if find a suitable upper level event then
            extract information out of RDS;
            build E;
            save E;
        end
    end
end
```

**Figure 8:** EDS Processing Algorithm

As stated above, a **rule** is an assignment specification mapping a raw data set to an event. It is comparable to a blueprint of an event. One rule can be used to create multiple similar structured events. Each rule consists of five parts described in the listing below:

- **Requirement**: This field describes which upper level event is required for the event produced within this rule. There are two different possible entries here. On the one hand, it is possible to state a required event, e.g. the event description. On the other hand, it is possible to use the wildcard entry *any* in order to accept any event with the correct level.
- **Source**: The value stated here describes where the raw data sets can be found used in combination with this rule.
- **Event level**: This value consists of an integer defining the level of the produced event. The event level field contains either 1, 2 or 3.
- **Event description**: This field contains the event description text. *SED x turned on* is an example event description text used in context of the later discussed PriSEMD prototype.
- **Device**: In this field the SED device name is stated. For example *[TV]UE40ES6300* is the device name of the Samsung UE40ES6300 STV.

All rules used in context of the implemented PriSEMD prototype can be found in the appendix (Table 24). Within the implementation, the fields *requirement* and *source* are divided into two fields each. The reason for this is explained in the implementation Section 4.2.4.

The immense effort spent on the event calculation procedure is done in order to create a flexible framework. This framework eases the inclusion of new raw data sets and allows therefore extending the set of measured interfaces. Extending PriSEMD for example with the measurement of a new interface leads to new raw data sets, events and rules.

### 3.1.3 Analyzing Phase

The Analyzing Phase is the third phase of the PriSEMD concept. The goal of this phase is to calculate facts concerning user behavior. As displayed in Figure 4, the input consists of the Processing Phase events and additional information, the output of user behavior related information. The additional information is not provided by PriSEMD and has to be requested of external services. A database or webpage has to be requested for example in order to receive additional information which is needed for meaningful information processing. Requesting additional information is optionally not mandatory.

An example where gathering additional information is required could be analyzing TV program events. In this particular case, an electronic program guide will be requested in order to obtain additional information about the TV program.

Another example where no additional information is required could be analyzing the power status events of a SED. The power status events are determined when a SED was turned on and turned off. A list of time periods a SED has been turned on could be one possible output of the Analyzing Phase.

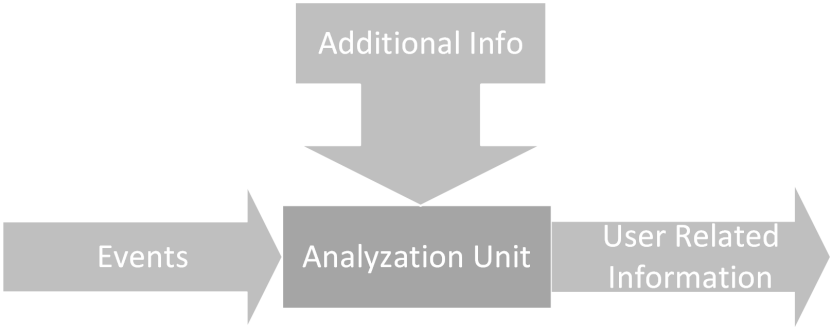



**Figure 9:** User Related Information Processing

### 3.1.4 Presentation Phase

The Presentation Phase is the last phase of the PriSEMD concept. The goal of this phase is to provide the results of the previous three phases to the two stakeholders - the local and third party user.

The accessible data within the presentation interface of PriSEMD is different for each stakeholder. The local user is able to view all data without any regulation or restriction. In contrast, the third party user has only the possibility to see data approved by the local user.

The local user can view all raw data sets produced within the Collection Phase, all events of the Processing Phase and all user behavior related data of the Analysis Phase. In contrast, the third party user can only view user behavior related data of the Analysis Phase.

Figure 10 illustrates the two stakeholders within a general interaction diagram. The local user is displayed on the left side. He has two different options while interacting with PriSEMD. First, he has the possibility to view the complete data produced by PriSEMD. Second, he has the opportunity to regulate the access to the PriSEMD data set for third party users. The third party user is shown on the right side of Figure 10. The only available interaction method for the third party user is viewing a restricted set of data which has to be first approved by the local user.

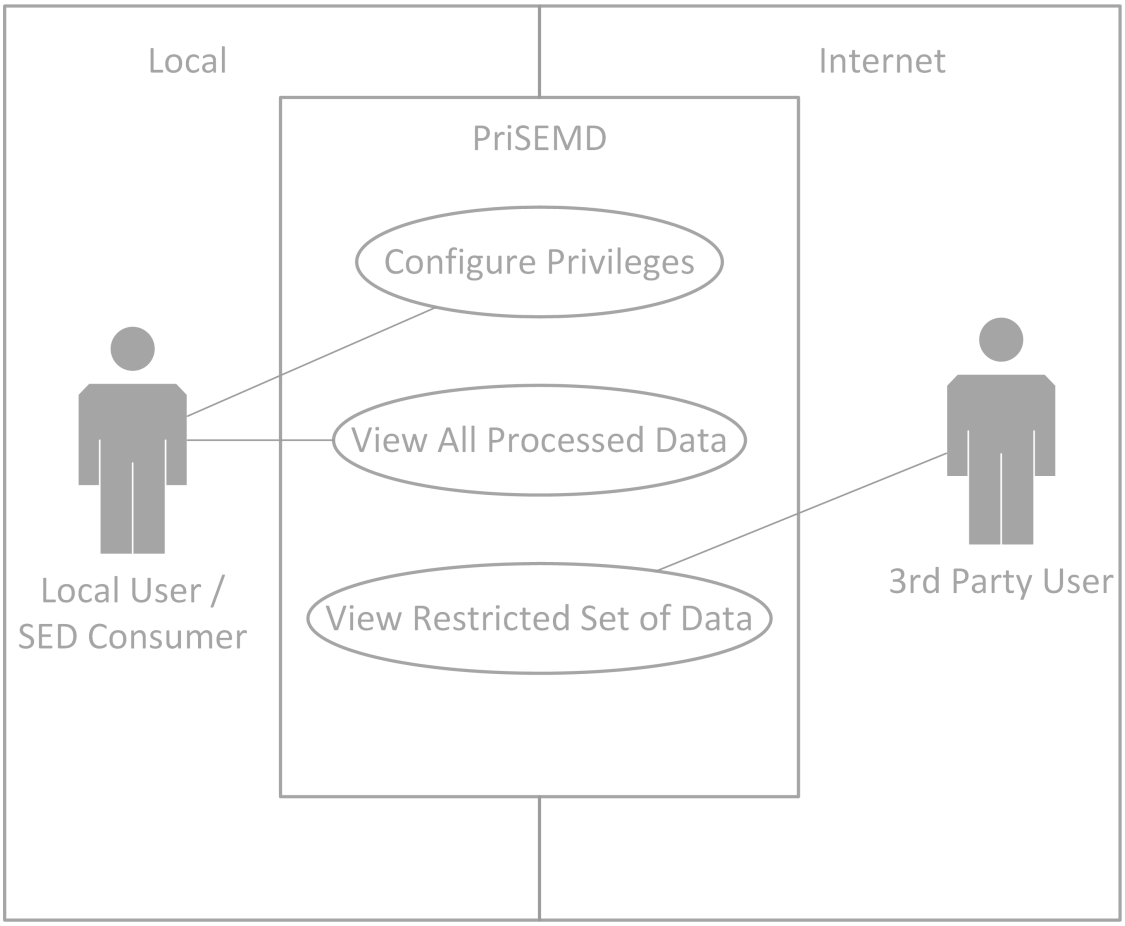



**Figure 10:** Stakeholders Using PriSEMD

#### Distinction Between Users Accessing the Presentation GUI

In this paragraph, we describe how PriSEMD detects which user tries to access the PriSEMD presentation layer. Therefore, roles are assigned to the users beforehand. Two categories of roles exist for this purpose, the **local user role** and **third party user roles**. Users assigned to the local user role are allowed to access all data produced by PriSEMD. In contrast, users that are assigned to the third party user role can access only approved data.

Thus, there are one local user role and $n$ different third party user roles. The third party user roles can be named differently and the access rights can be managed individually. A third party user role can e.g. be created for a specific TV station. It is possible to assign rights to access only data belonging to the TV program or which channel was watched at a specific time. Other data is not accessible for a user with this particular role.

The role validation depends on the network access method, either local or remote. Each user connecting to PriSEMD from the local network is assigned the role of a local user. After entering the valid local user password, the presentation of PriSEMD can be accessed. The required password is the same for all local users in order to keep the access procedure as simple as possible.

All users connecting to PrisEMD remotely, e.g. from the Internet, have to pass a login dialog. For the login procedure previous defined username and password combinations are required. PriSEMD is able to verify the user depending on the used username and can assign an appropriate role to it. Both local and third party user can use the remote access method. Any connection to the PriSEMD device has to be encrypted in order to preserve confidentiality.

**Regulation of the PriSEMD data**

As stated above, the accessible data depends on the assigned user role. A user with assigned local user role is allowed to access all data, a user with assigned third party user role can only see previous approved data sets. Within this paragraph, it is outlined how PriSEMD manages the third party user data access and how the approved data is tagged.

The data sets are tagged with so called privileges. A **privilege** is a label assigned to both data set and third party user role. A third party user is allowed to access a data set if the third party user role and the data set are tagged with the same privilege.

Table 2 shows several examples. Each line of the table represents an entity of the PriSEMD system. The first four entries are users, the last four are events. Users *a* to *c* have been with assigned third party user role, user *l* has been assigned to the local user role. User *a* got the tag *x* and *y* assigned. Therefore, he is able to see event 1 and event 2, because event 1 is labeled with tag *x* and event 2 with tag *y*. User *b* is able to see event 2 and event 4, User *c* can only see event 1. Event 3 has no tag assignment. That means, it is invisible to any third party user. Only the local user is able to see event 3. Due to the fact that the local user is able to view all data in an unrestricted way he does not need any tag assignment.

| Entity | User Role | Privileges |
|---|---|---|
| <user *a*> | third party user role | <privilege *x*>, <privilege *y*> |
| <user *b*> | third party user role | <privilege *y*>, <privilege *z*> |
| <user *c*> | third party user role | <privilege *x*>, <privilege *w*> |
| <user *l*> | local user role | (no privilege assignment required) |
| <event 1> | | <privilege *x*>, <privilege *w*> |
| <event 2> | | <privilege *y*> |
| <event 3> | | (no privilege assignment) |
| <event 4> | | <privilege *z*> |

**Table 2:** Privilege System Example

## 3.2 Meterable Data - Processable Information

An overview about SED data measurement is given in this section. It is discussed which SED interfaces can be measured, which data can be gathered while analyzing a specific interface and which information can be extracted out of the measured data.

As outlined in Chapter 2, SEDs have various interfaces where data measurement can be implemented. The measurement of three concrete interfaces is outlined in this chapter, analyzing network traffic, using a specific SED network service (UPnP) and recording infrared signals from remote controls. All three interfaces are selected considering the aspects mentioned:

1. The technical effort for establishing the metering procedure has to be as low as possible.
2. The gathered data of the interface has to be useful for extracting user behavior related information.
3. The interface has to be widely accepted and multiple SEDs have to have the interface integrated.

The goal of the SED data measurement is to collect data about the environment where the SEDs are used. This data should consist of information about user habits and behavior, and about details of the SEDs used. Analyzing STV program data can be used for example to describe the viewing behavior of a STV user. Collected UPnP data consists of device specific details like SEDs' device name, model number and manufacturer name.

### 3.2.1 Network Analysis

Network analysis is the process of reading and analyzing network traffic between SEDs and other network devices located in the Local Area Network(LAN) and Internet.

This type of measuring was chosen, because a network interface is implemented in nearly every SED. The technical effort to establish network analysis is low. As mentioned in Chapter 3.1, a network infrastructure has to be set up as a

requirement for PriSEMD. The analysis device has to be implemented in the same network as the SEDs. The best attempt to implement the analysis device is to set it up as a **transparent proxy**. A proxy is an instance working as an intermediary between two sides. A transparent proxy is further an intermediary working without the notice of the two parties [38]. As shown in Figure 3, the network implementation of PriSEMD is done as a transparent proxy. The analysis of the network traffic is possible, because it passes through PriSEMD.

Network traffic consists of network packets sent from devices within a network. Every network packet has to be analyzed using a technique called Deep Packet Inspection (DPI). DPI means analyzing a network packet in detail and looking into the various protocol parts of a packet in order to extract desired data. Each protocol part consists of a header and a payload section, which are both considered by DPI. DPI makes it possible to gather all available details of the read network packets. Each network packet is built of different protocol layers. A description often referenced of these layers is the Open System Interconnection (OSI) model. The layer names of the OSI model are used in this subsection [48, p. 41].

Several aspects of network packets are described in the following paragraphs. Each paragraph outlines the measurable data and the processable information.

#### MAC Address

**MAC Address - Technical Description**

The Media Access Control (MAC) address is used within the medium access control layer, a sub layer of the OSI models data link layer. The address is used to identify a device in a network, especially in LANs. Each MAC address is 48 bits long and can be divided into two parts. The first part, often called prefix, is a vendor specific identifier. The MAC prefixes are administrated by the Institute of Electrical and Electronics Engineers (IEEE) which sells prefixes to interested companies. The second part of a MAC address is device specific and assigned by company. [48, p. 282ff][18].

IEEE provides a service[a] that shows the manufacturer of a network device. The information usually consists of name and address of the network device manufacturer. It can be requested while providing the MAC prefix. An example is shown in Figure 11. The test device used in this work is a Smart TV of the manufacturer Samsung. The STV has the MAC address *48:44:f7:a7:66:f8*. As mentioned above, the MAC address can be divided into two different parts. A query at the IEEE service delivers the information that the manufacturers prefix *48:44:f7* belongs to „Samsung Electronics Co., LTD“. In addition, the IEEE service provides the postal address of the manufacturers headquarter.

[a] `http://standards.ieee.org/develop/regauth/oui/public.html/`


**Figure 11:** MAC Address Example of Samsung STV

The following listing shows for which purposes the MAC address can be used for:

1. **Obtain device manufacturer details**: However, an important problem in this context is that the manufacturer of the SED and the manufacturer of the network interface could be different. Assuming that SED *s* is manufactured by company *c1* and has integrated a Wi-Fi network card from manufacturer *c2*. The MAC address is usually connected to the network hardware used. Due to the fact that the network device is manufactured by another company than the SED, it is probable that the MAC address prefix belongs to *c2*.

2. **Use MAC address as identifier**: A MAC address can be used as an identifier for a SED. As a rule, a device should have a unique MAC. However, MAC spoofing is a well-known technique to fake MAC addresses. It is possible to gain root access on SEDs (for example a Sony Bravia STV with rooted Linux OS [3]) and change the MAC address using root rights [23]. Normally, the MAC address is not changed by the user and can therefore be used as an identifier.

#### IP Protocol

**IP - Technical Description**

The Internet Protocol (IP) is a computer network protocol operating at the network layer. The purpose of the network layer is to bring packets from source to destination. This layer is the first of the OSI model implementing

an end to end transmission. The IP protocol is currently (2014) available in two different versions, IPv4 and IPv6. IPv4 is focused in this work, because it is most wide spread and all devices support it [47, p. 367]. The two most common Transport Layer Protocols based on IP are TCP and UDP. On the one hand, the Transmission Control Protocol (TCP) is a connection-orientated protocol supporting techniques to deliver packets without packet loss. On the other hand, the User Datagram Protocol (UDP) is a connection-less protocol with a smaller overhead than TCP where packet loss is possible.

The analysis of a network packet's Internet Protocol(IP) section provides access to the IP address of communication source and destination. Depending on the sending direction either the source IP or the destination IP belongs to the SED. The other address references an entity the SED is communicating with.

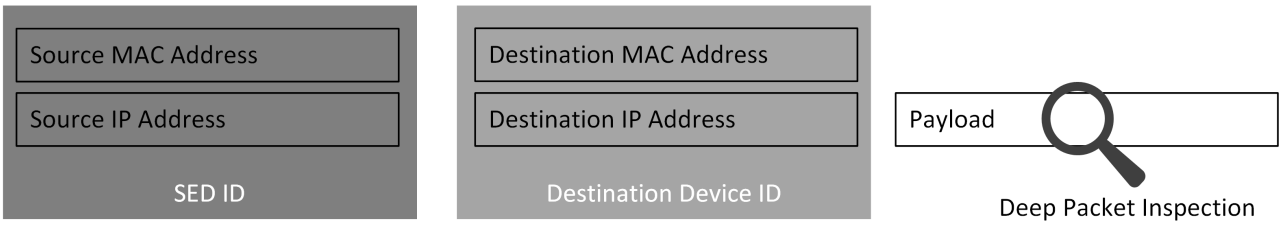



**Figure 12:** Building IDs from MAC and IP Addresses

Figure 12 shows a network packet sent from a SED. The source IP address belongs to the SED device and the destination IP address to the communication partner. The combination of both, IP and MAC address can be used to create a unique identifier. The payload of the IP packet is analyzed using DPI.

#### HTTP Protocol

**HTTP - Technical Description**

The Hypertext Transfer Protocol (HTTP) is a computer network protocol of the application layer. HTTP is a stateless request response protocol using TCP and IP on the underlying layers. There are two different groups of HTTP messages, the HTTP Request and HTTP Response messages. HTTP Request messages are sent from client to server in order to request information. HTTP Response messages are sent from server to client as an answer. A client can for example request data from a web server and the web server returns a response message including the requested information. [48, p. 683] [20]

The HTTP section structure is explained hereafter using the example showing in Figures 13 and 14. The structure of an **HTTP Request message** is illustrated in Figure 13. The HTTP Request message consists of a first line and a header section. The first line contains the message type, a part of the requesting URL and the HTTP protocol version. The header section is a list of headers which are specified as *headername:value*.

**HTTP Request Message:**

```
GET /content/25-09-89/cake.png HTTP/1.1\r\n
Host: my_website.com\r\n
Connection: keep-alive\r\n
Cache-Control: max-age=0\r\n
User-Agent: Mozilla/5.0 (Windows NT 5.1)\r\n
Accept: */*\r\n
\r\n
```

**Figure 13:** HTTP Request Structure Example

The structure of a **HTTP Response message** is outlined in Figure 14. The HTTP Response message consists of a first line, a header and a body section. The first line contains the used HTTP protocol version and a server status code. The header section is similar structured to the section of the HTTP Request message. The body section contains the payload information requested by the client. In Figure 14, the example body of the HTTP Request message contains HTML code [20].

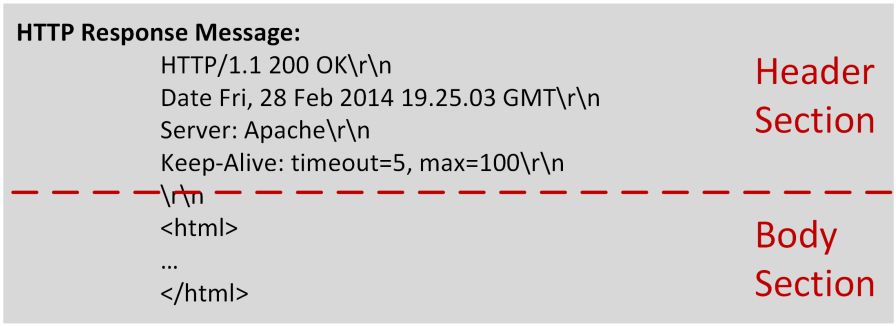



**Figure 14:** HTTP Response Structure Example

A lot of SEDs are using HTTP to load content from web servers, e.g. integrated web browsers of STVs or video gaming consoles. The listing below shows two different examples which information can be extracted while analyzing HTTP messages:

1. **Request URL:** The request URL can be extracted from HTTP Request messages. In order to obtain the complete URL, it is necessary to combine the *Host* header field and the URL part from the first line of the message. As shown in Figure 13, the complete request URL is *http://www.my_website.com/content/25-09-89/cake.png*. In the context of analyzing SEDS, request URLs can be metered in order to track visited websites. Each metered URL represents a link to a website visited by a SED. Using this information, a protocol of visited websites can be created. This information can be used to describe the preferred websites of a user.

2. **Response Content:** The response content can be extracted while analyzing HTTP Response messages. In Figure 14, the content consists of HTML data. One possible analysis strategy can be searching for key words. Searching for a key word like *soccer* or *football* can classify for example the requested web page to the category *sport*.

#### HTTPS and DNS

**HTTPS - Technical Description**

The Secure HTTP (HTTPS) protocol is an extension of the previous described HTTP protocol. Between Transport and Application Layer another layer is inserted. This layer uses Transport Layer Security (TLS) or Secure Socket Layer (SSL) in order establish a secure connection between both communication partners. The result is that the HTTPS protocol section is encrypted and integrity-safe [48, pp. 853-854].

**DNS - Technical Description**

The Domain Name System (DNS) is a computer network protocol of the application layer. DNS allows it to map IP addresses to human readable names, for example the URL of a webpage. DNS server provide a service that can be requested with either an IP address or a domain name in order to resolve this mapping [48, pp. 611-612].

Server Name Indication (SNI) allows one server to host multiple websites with different domains [52]. Analyzing HTTPS packets provides information about the requested server, but not about the domain name, because HTTPS messages are encrypted. However, analyzing HTTPS packets in combination with DNS requests allows gathering the domain name indicating the particular requested website on the server.

Figure 15 shows an example how this analysis could be executed. Due to the fact that HTTPS packets are encrypted, it is not possible to extract information of the HTTPS protocol section. The IP section is not encrypted and the IP address of the source can be found in plain text. The IP address of the source device in Figure 15 is for example *146.134.201.001*.

Recorded DNS messages contain the IP address or the requested domain name of a web server. This information can be collected in order to track the web sites requested from SEDs. Analyzing DNS requests made in combination with the HTTPS requests allows finding out the domain name of the source device. This is possible, because DNS packets are not encrypted. The requested domain name in Figure 15 is *www.sport_website.de*. The measurement mechanism described here is not flawless. A DNS look-up could e.g. be made before measurement was started. In this particular case, no DNS messages are received by the measurement process.

#### SSDP Protocol

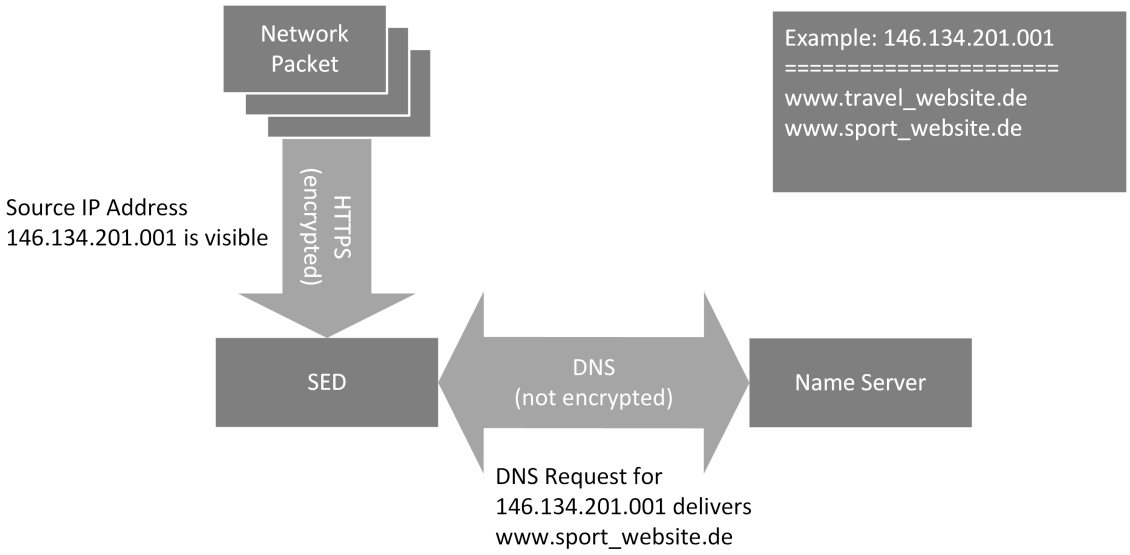



**Figure 15:** Analyzing HTTPS and DNS Traffic

**SSDP - Technical Description**

The Simple Service Discovery Protocol (SSDP) is a network protocol running at the application layer of the OSI model. The SSDP message structure is very similar to HTTP messages. Each message consists of a start line and a header section with several header value pairs. In contrast to HTTP, a SSDP message has no body section. The start line consists of exactly one line describing the SSDP message type. SSDP uses UDP and IP in the underlying protocol layers [22].

SSDP usually occurs in combination with UPnP which is explained in Chapter 3.2.2. Only network relevant aspects of the SSDP protocol are mentioned in this section. SSDP messages are sent out from a source device as a broadcast. It supports five different message types. Each message type has a specific header value defining the type of the message. The listing below shows the possible SSDP message types:

- **ssdp:alive** is sent from a device using multicast in order to announce other network devices that it is available. A device usually sends this message when it is turned on or is newly connected to a network. After sending this initial message, a device sends ssdp:alive messages in a specific interval to renew its announcement. The interval of sending depends on the device's SSDP implementation. It can vary from device to device.
- **ssdp:byebye** is sent from a device using multicast in order to announce other network devices that it is no longer available. A device usually sends this message when it is turned off.
- **ssdp:update** is sent to announce change of a specific device ID to other devices.
- **ssdp:discover** is sent to search for other devices in the network using multicast.
- **ssdp:all** is sent as an answer to a previous ssdp:discover. A device receiving a ssdp:discover responds with a ssdp:all message send to the source of the ssdp:discover.

The **ssdp:alive** and **ssdp:byebye** messages can be used to track the on/off status of a device. Each device using SSDP publishes its current status to all devices in the network. Therefore, a broadcast message with a corresponding header is transmitted. Receiving a SSDP packet with the header field „ssdp:alive“ indicates that the SED is running; it is halted after receiving a packet with „ssdp:byebye“. A detailed concept of extracting the device status information is described in Chapter 4.2.4.

The **ssdp:discover** and **ssdp:all** can be used to actively gather information about devices in a network. After performing a *ssdp:discover* all available and SSDP using devices responding with a *ssdp:all* message. The analysis of *ssdp:all* messages leads to a list of all available devices using SSDP.

One problem of SSDP is going along with the used transport layer protocol UDP. UDP messages are delivered and sent without connection handling. This leads to the fact that there is no guarantee that a message arrives. A reasonable certainty that a message arrives can be achieved while sending a message multiple times.

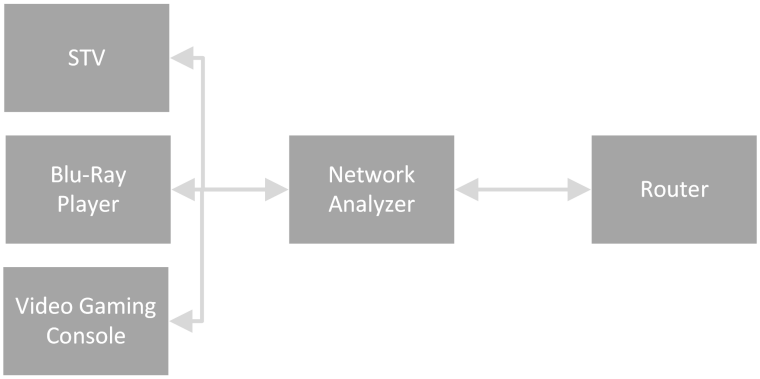



**Figure 16:** SSDP Example Environment

Three different SEDs are displayed in Figure 16 - a STV, Blu-Ray player and video gaming console. It is assumed, that all SEDs are using SSDP. The network analyzer is the entity which analyzes the network traffic in order to obtain facts about the connected SEDs. All SED devices are initially turned off. The first turned on device is the STV. After turning on the STV, the network analyzer receives *ssdp:alive* messages from it. Turning on the Blu-Ray player and the video gaming console leads to the same result, the network analyzer receives *ssdp:alive* messages from both devices. Turning off the Blu-Ray player afterwards, the network analyzer receives *ssdp:byebye* messages from this device. Due to the fact that the other two devices are still turned on, they are sending *ssdp:alive* messages.

In the next step, a *ssdp:discovery* is performed by the network analyzer. The Blu-Ray player and the video gaming console are responding with *ssdp:all* messages. The STV does not respond, because it is turned off.

#### DHCP

**DHCP - Technical Description**

The Dynamic Host Configuration Protocol (DHCP) is a computer network protocol of the application layer. The main functionality of DHCP is the assignment of network addresses to network devices. The DHCP server is responsible for the assignment and communicates therefore with each client in the network. A network address is assigned to a device for example when the device is turned active or newly joining the network. DHCP is based on UDP and IP [12].

DHCP can be used to achieve information about the on/off status of a network device. Analyzing this protocol can be an alternative to analyzing SSDP for example because the device does not support SSDP.

In order to analyze the on/off status three different messages types of the DHCP protocol have to be distinguished:

1. **DHCPDISCOVER** is sent from a client as a broadcast message to all network devices available in order to locate a DHCP server.
2. **DHCPOFFER** is sent from the DHCP server to a client in order to response to a previous send DHCPDISCOVER message. The DHCPOFFER message contains several parameters.
3. **DHCPRELEASE** is sent from a client to a known server, because the client wants to terminate the network connection.

The active status of a SED can be metered while analyzing DHCPDISCOVER messages. Metering the offline status is more complicated. The DHCPRELEASE packet is not always sent by a SED. It depends, whether the SED network implementation sends the message or not [12].

#### HbbTV - An Example for a Measurable Network Service

**HbbTV - Technical Description**

Hybrid Broadcast Broadband TV (HbbTV) is a technology used by STVs. HbbTV combines DVB reception and the Internet to offer users more information about the currently running program. HbbTV uses HTTP messages to transmit this information. Figure 17 shows the conceptual communication procedure. The STV requests for additional TV program content and the broadcasting station delivers the desired information [25].

The requests sent to the TV station can be categorized in two groups, start-up requests and periodic requests. A **start-up request** is executed when a new channel is selected by the user. If HbbTV is supported by the selected

channel, a URL is provided. After calling this URL, the STV shows a notification that HbbTV is available. Is HbbTV not provided by the TV station, no URL is provided and the STV does not perform the call. **Periodic requests** are sent after the start-up requests in periodic time intervals [26].

Not all TV stations and STVs are supporting HbbTV. At the beginning of 2014, there are about 60 channels in Germany supporting HbbTV. However, the list of supporting channels is growing [28]. During the last two years, a lot of new STVs using HbbTV have been deployed. Furthermore, it is assumed that the number of supporting STV devices is still growing [33]. A specification of HbbTV can be found in [15]

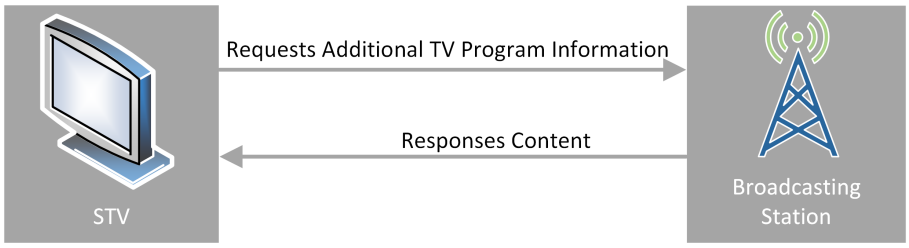



**Figure 17:** HbbTV Conceptual Communication Procedure

Analyzing HbbTV is an example of measuring a specific service within the network traffic. The listing below shows two possible information aspects that can be extracted while analyzing HbbTV traffic. Due to the fact that HbbTV uses HTTP packets, the analysis of HbbTV is similar to analyzing HTTP packets:

- **URL:** The URL of the start-up request usually includes the name of the current running STV channel. The HTTP Request is sent directly after switching over to the channel. This leads to the fact that the combination of channel name and timestamp can be used to state which channel is watched at a specific time. A STV sends e.g. a request with the URL *http://hbbtv.prosieben.de* at the time *2014-08-18 14:58:01*. This information can be compared to the fact that the channel *ProSieben* was turned on at *2014-08-18 14:58:01*.

- **Content Analysis:** Analyzing the content of a HbbTV HTTP Response is similar to analyzing a general HTTP Response message. Mostly, the content of HbbTV consists of HTML code which can be parsed and evaluated for further information gathering.

### 3.2.2 UPNP - A SED Provided Service Example

The idea of the Universal Plug and Play (UPnP) technology is to connect devices in a LAN in order to provide a communication framework. The established communication framework allows all devices supporting UPnP to communicate with each other. Each device using UPnP offers services that can be requested by other devices. UPnP is defined as a universal standard and therefore non-proprietary. In other words, UPnP devices are able to communicate with each other regardless of whether being from the same manufacturer or not. UPnP was founded by Microsoft in 1999 and is now administrated by a multi-company-initiative called Digital Living Network Alliance(DLNA). UPnP is implemented in a lot of different devices, for example in smart phones, tablets, notebooks, routers and SEDs [53, pp. 135-138][21, 37].

After outlining the structure and functionalities of UPnP an analysis of gatherable data is given afterwards.

#### UPnP Structure

In a UPnP environment, two categories of actors are differentiated - UPnP devices and UPnP control points. A **UPnP device** provides one or more services that can be requested by other devices using UPnP. Requesting information about the UPnP device or triggering a specific functionality can be e.g. realized with a service. An **UPnP control point** is an entity which can request UPnP services. The requesting procedure is handled while exchanging messages. A request message sent to a UPnP device triggers a specific service and a response message is sent back to the UPnP control point. A SED can be a UPnP device, a UPnP control point or either both of them at the same time.

The functionalities defined in the UPnP specification[4] are displayed in Figure 18. The boxes on top of the figure represent the five different UPnP functionalities. The boxes below show details of the network protocols used to implement the specific UPnP functionality [37, 22]. Each functionality is described in the following listing:

[3] based on: http://de.wikipedia.org/wiki/Universal_Plug_and_Play#mediaviewer/Datei:Upnp_architecture.svg

[4] http://upnp.org/sdcps-and-certification/standards/

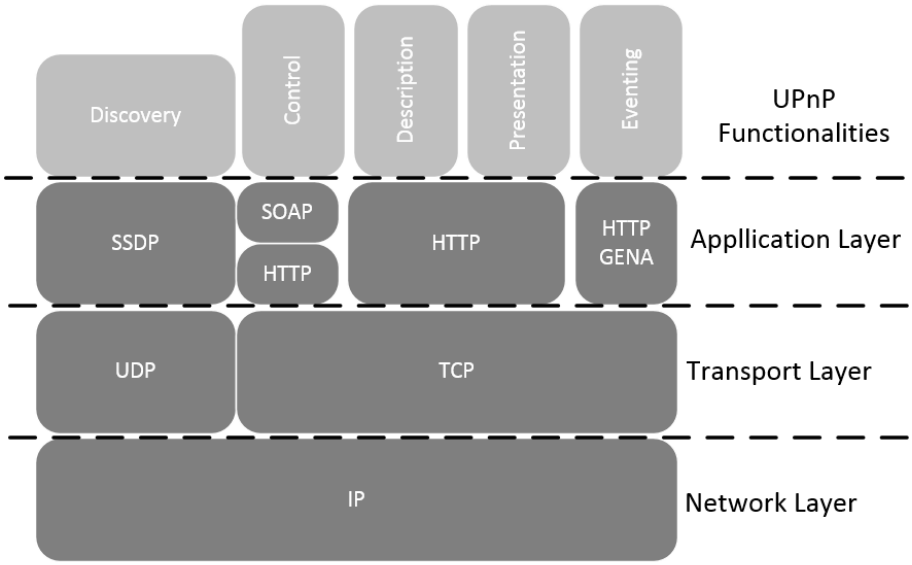



**Figure 18:** UPnP Architecture[3]

- **Discovery**: The UPnP Discovery functionality is used by both, UPnP device and control point when they are newly connected to a network. An UPnP device uses the Discovery functionality to announce its' provided services to other devices in the network. A UPnP control point uses this functionality to search for other UPnP devices. In both cases, a discovery message is sent as broadcast to all available network devices. The discovery message is based on the SSDP protocol. A description of the SSDP protocol structure was given in Section 3.2.1.

- **Description** The UPnP Description is a topic for characterizing services and functionalities provided by a device. A XML based file provides all necessary description details needed for requesting a service. The file contains a list of all services provided by a device, URLs where the services can be requested and manufacturer details. A reference to the location where the description file can be found is delivered with each discovery message sent.

- **Control** UPnP Control is a mechanism allowing a control point to interact with another device. After receiving the link to the detailed service description, a control point is able to request the services described within the description. For this purpose, a control point has to send an appropriate message. This message is formatted in XML and delivered using the SOAP protocol ( see Figure 18). The concrete structure of this message can be found in the service description. The response message sent from the device to control point is delivered using the same procedure. Within a service request, it is possible to provide service specific values, like e.g. input parameters. Service responses can further provide return values, like e.g. results.

- **Eventing** The UPnP Eventing is a publish subscribe mechanism for service variables. A control point can subscribe to a service variable. Every time the variable is changed, the device publishes the new value and the control point receives the updated value automatically. The General Event Notification Architecture (GENA) is the used protocol for this functionality. Further information about the protocol can be found in the corresponding IETF draft document [30].

- **Presentation** The UPnP Presentation provides a presentation web page for controlling a device. For this purpose, a device forwards a URL pointing to the specific page which can be inspected by a web browser.

#### UPnP Data Measuring of SEDs

Data measurement can be realized while requesting available UPnP services of SEDs. Therefore, an appropriate request message has to be sent to the UPnP device. The response message contains the desired information.

An example using the measurement procedure mentioned above is illustrated in Figure 19. A hypothetical service called *GetCurrentSongName* is provided by a SED sound system. After requesting the *GetCurrentSongName* service with a request message, a response is sent back by the sound system. The response message contains the name of the current running song on the sound system.

[5] source of smart sound system image: [44]

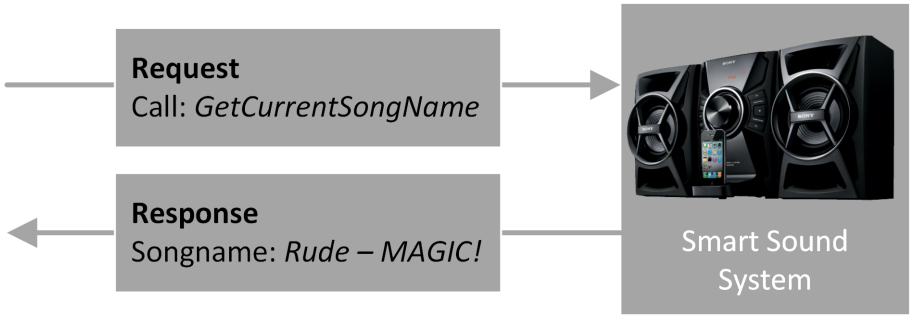



**Figure 19:** UPnP Example[5]

The UPnP metering procedure has one disadvantage. The data provided by the services is depending on which device is used. Due to the fact that the service offered is not standardized, each manufacturer can implement services on his own. The provided and available data can vary from manufacturer to manufacturer and even within devices from the same manufacturer. The variation of services leads to the fact that it is very difficult to establish a general UPnP metering functionality. Each device or model series has to be analyzed on its own.

### 3.2.3 Infrared

The measurement of infrared (IR) signals is topic of this section. In the context of SEDs infrared technology is usually used by remote controls. Remote controls send control messages to a SED. Sending a turning on or off signal to the SED leads for example to an active or inactive state of the device. A lot of SEDs are using remote controls in order to let the user interact with the SED. STVs, disc players, sound systems and set top boxes are for example using them.

The way of transmitting signals is unidirectional. Remote controls send signals only. They do not receive anything. The communication flows from remote control to receiving device. The receiving device has to have an infrared sensor in order to be able to receive the signals.

IR signals can be measured while establishing an analysis device with implemented IR sensor near to the SED device. When a remote control sends IR signals to the SED, the analysis device receives the signal as well, because IR signals are sent as broadcast. The recorded infrared signals can be used for two purposes:

1. **User Activity**: Received remote control signals are a sign for user activity. Each time the sensor receives signals, the user has pressed a button on the remote control.

2. **Pressed Button Name**: Infrared signals can be analyzed in order to extract the name of the pressed remote control button. The extracted button name can be used to determine which action was performed by the user. In addition, it is possible to track the using behavior. Measuring a STV and receiving signals from the *channel down* button leads for example to the fact that the user has switched to another channel. Or receiving signals from the *volume down* button might be an indication of pulling the volume down, because e.g. television advertising is currently running on the STV.

One problem while working with remote controls is that an own remote control is available for each device. Across manufacturers, there are a lot of different remote controls available. Furthermore, the signals are quite arbitrarily selected. A standardization of signal codes is not available; each manufacturer implements codes on his own. This leads to the fact that each remote control has to be considered individually.

## 3.3 Privacy Aspects

The privacy aspects concerning PriSEMD data measurement are discussed in this subsection. The main focus is on the privacy of the data which is gathered, presented and forwarded to others. After attempting to make a definition of the term privacy in the context of PriSEMD, the stakeholder interests of local and third party user are outlined. Attack vectors and possible solutions for preventing these attacks are discussed afterwards. In the end, a concept for anonymously transferring data to third party users is introduced.

As Jon Kleinberg et al points out [31], **privacy** is „notoriously hard to define and capture rigorously“. Many different definitions of privacy are available. A definition from an English dictionary says [11]: Privacy is „a state in which one is not observed or disturbed by other people“. A German definition of privacy reads as follows [16, p. 5]: „Privacy/Data Protection is a person's ability to control own personal data“.

A combination of both definitions is considered in this thesis. On the one side, the aspect that people should not be observed or disturbed, and on the other side, that people should have control over their personal data.

### 3.3.1 Different User Interests

As mentioned at the beginning of Chapter 3, two different stakeholders are present, the local and the third party user. Both users have different interests concerning the PriSEMD provided data.

**Third Party User**

Data is presented to the third party user through the user interface of the *Presentation Phase* (see Chapter 3.1.4). The third party user is interested in using the data for different purposes, for example for continuous product improvement, further developing and optimizing services or personalization of advertisement.

In order to fulfill these purposes, the third party user wants to have the data as detailed as possible. In the context of PriSEMD, detailed data characterizes everything measured within PriSEMD, the raw data, events and the user behavior related data. From the perspective of the third party users, the dissemination of data shall be without restrictions. In other words, all measured data shall be available for each third party user.

An example is the personalization of advertisement mentioned above. The adaption to the target's interests and habits increases the revenue of a specific advert. Refining personalized advertisement can be accomplished by using detailed information about the user. The level of detail is decisive for how precisely the advertisement fits to a person. In addition, more details can help improving products or services, because the manufacturer obtains detailed information about the usage and can determine present problems.

**Local User**

In contrast to the interest of the third party user, there are the interests of the local user. The interests of the local user cannot specifically be determined as the interest of the third party user.

Many users do not perceive that SEDs are sending data to third parties. On the one hand, this could be caused by a lack of understanding Internet and computer network technologies. These users do not understand what effects are going along with technologies concerning data transmission. Connecting a SED to the Internet allows the device for example to communicate with other devices, even remote devices. This effect of not perceiving what is happening is reinforced by the fact that data transmission is executed under the surface. The user interface does generally neither show which data is transmitted to whom, nor which data is transmitted at all.

Besides not perceiving data transmission, the users are unaware of the consequences. Nearly every data produced by the SED can be transmitted to others providing that the device is connected to a network. A STV can for example measure when a specific program is watched and how long the device is running each day. This data can be transmitted to the manufacturer or to advertising analysts in order to create a profile and track users' behavior.

The concept of PriSEMD should give local users the possibility to look behind the surface of SED data transmission. Using the PriSEMD presentation, the local user has the possibility to get an overview about transmitted data of all analyzed SEDs. Furthermore, an overview is provided for which purposes the transmitted data can be used for.

The main task of PriSEMD is to sensitize the local user concerning SED data transmission. After inspecting the PriSEMD presentation the local user is able to answer the following questions:

- Is data sent to third or illegitimate parties?
- Which data is disseminated, which not?
- Does the disseminated data violate my privacy?
- Which conclusions to user behavior and habits can be drawn by the disseminated data?

However, data dissemination is not at all against the user's interests. A benefit for the user could be transmitted data that is used to improve products or services. A new set-top box is for example available, but the operating system has several bugs concerning the user interface. Fixing these bugs and having a patch as quickly as possible is in the interest of the local user.

**Tradeoff Between Third Party and Local User**

The interests of third party and local users are contrary. It is not possible to cope with both interests while neglecting each other. A benefit for the one is a disadvantage for the other. Dissemination of detailed private information is a benefit for the third party user, while it could be negative for the interest of the local user. Lower the detail of data is a disadvantage for the third party user, but can be positive for the privacy efforts of the local user. The lower the level of detail, the less the value for the third party user. Figure 20 illustrates the conflict of interest between both stakeholders.

A tradeoff between the interests of local and third party user has to be made in order to solve the conflict of interest. This tradeoff has to consider on the one side the wished level of details of the data and on the other side the privacy debts of the local user.

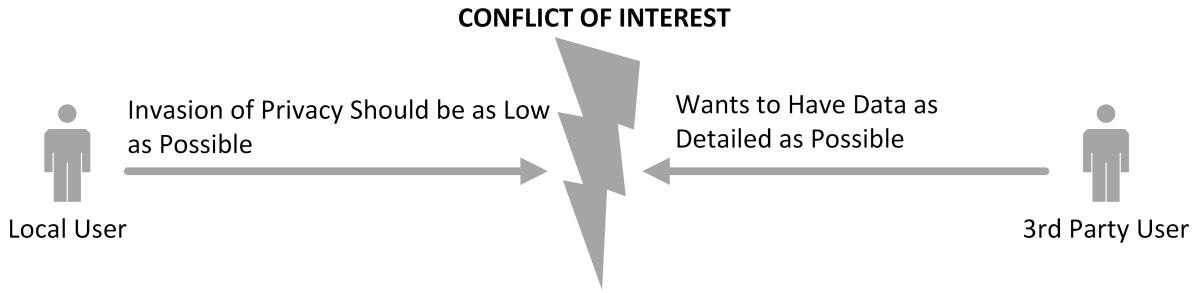



**Figure 20:** Conflict of Interest Between Local and Third Party User

### 3.3.2 Attack Vectors and Possible Protection Mechanisms

Four different attack vectors are outlined in this subsection. Each attack vector is a possible attack to the privacy of a local user. The goal of this subsection is to show why it is important to establish privacy enhancing technologies (PETs) in order to protect the privacy of the local user.

**Attack Vector: Local User Profiling**

Profiling means gathering characteristics of a person such as habits, behavior and interests. A set of user behavior related data has to be analyzed and processed in order to improve this profile. After processing, the profile can be used to describe personal details of the appropriate person.

In the context of PriSEMD, the third party user is able to create a profile from the local user by analyzing the provided data. Provided data about the amount of running SEDs in the local users environment can be used to determine if the user is interested in technology. User $x$ has for example only a STV running in his environment and user $y$ has a STV, a video gaming console, a smart set top box and several computer devices. Within a profiling procedure user $y$ will be classified as interested in technology, because he uses a lot of different smart devices. In contrast to user $y$, user $x$ will be classified as not interested in technology.

As stated in [14], it is impossible to prevent absolute disclosure of data while providing data. However, it is possible to provide detailed information while ensuring a high level of privacy. The mechanism needed is called „differential privacy“. C. Dwork describes „differential privacy“ as a manipulation of data [14].

**Attack Vector: Confidentiality - Access to PriSEMD Data**

The access to PriSEMD data should be limited to approved third party users and denied to not known users. One solution to ensure confidentiality is presented in Chapter 3.1.4. Different roles are classifying the user in local or third party user. The privileges connected with the roles ensure that the user can access only approved content. A role based access control (RBAC) system checks that each user can access the content that is intended for him/her. For this purpose, the RBAC system observes the privileges connected with the roles [17, pp. 272-273]. A login system for accessing the PriSEMD presentation layer could be used to identify the user and map a predefined role to him/her.

Furthermore, transmission path between local and third party user shall be saved using encryption in order to ensure the confidentiality of the transmitted data.

**Attack Vector: Determining the Absence of a Local User**

PriSEMD data can be used to determine if a local user is absence or not. The absence of a local user describes if he is at home. A user can be for example on vacation. A third party user can use the absence information to break in the local user's home.

The determination depends on the accessibility of the PriSEMD data. If the data is disseminated in real time, the third party user is able to determine the absence in real time.

A solution to fix this dilemma is described by C. Dwork in paper [14]. To each transmitted data set $D$ a random noise value $n$ is added. The result data set $D_n$ is transmitted to the third party users.

$$D + n \rightarrow D_n$$

Concerning TV program this solution is problematic. If a high amount of data is collected, the third party is able to calculate the noise value. This is possible, because TV program is watched frequently. An example for a frequently running TV program in Germany is e.g. soccer on Saturday afternoons.

A better solution is to collect data and disseminate it after a period of time, not in real time. The determination of absence is therefore still being possible. However, the information cannot be used for burglars and is therefore less problematic facing the attack vector discussed here.

**Attack Vector: Attribution of Data to a Person**

PriSEMD data can be attributed to a specific local user or household. Characteristics in the PriSEMD data can be an indicator for that. For example, patterns in using SEDs in a particular manner. If a local user uses a specific combination of SEDs at a specific time each week, this information can be used to track the user.

One solution can be mixing identities while sending data to others and transfer the mixed data to third party user. A possible attempt is discussed in the paper of Drosatos et al [13].

### 3.3.3 Anonymous Data Transmission Concept

The anonymous data transmission concept is a PET for transmitting PriSEMD measured data anonymously to third parties. Anonymous transmission means that it is not possible for third parties to reconstruct with reasonable effort which data comes from which PriSEMD device and therefore from a specific household. After outlining the general concept, a detailed example is discussed.

**General Concept**

Figure 21 illustrates the general concept for anonymous data transmission. The central element of the concept is the **PriSEMD Broker**. *n* PriSEMD environments are connected with the PriSEMD Broker and each PriSEMD environment consists further of *x* SEDs connected with a PriSEMD device.

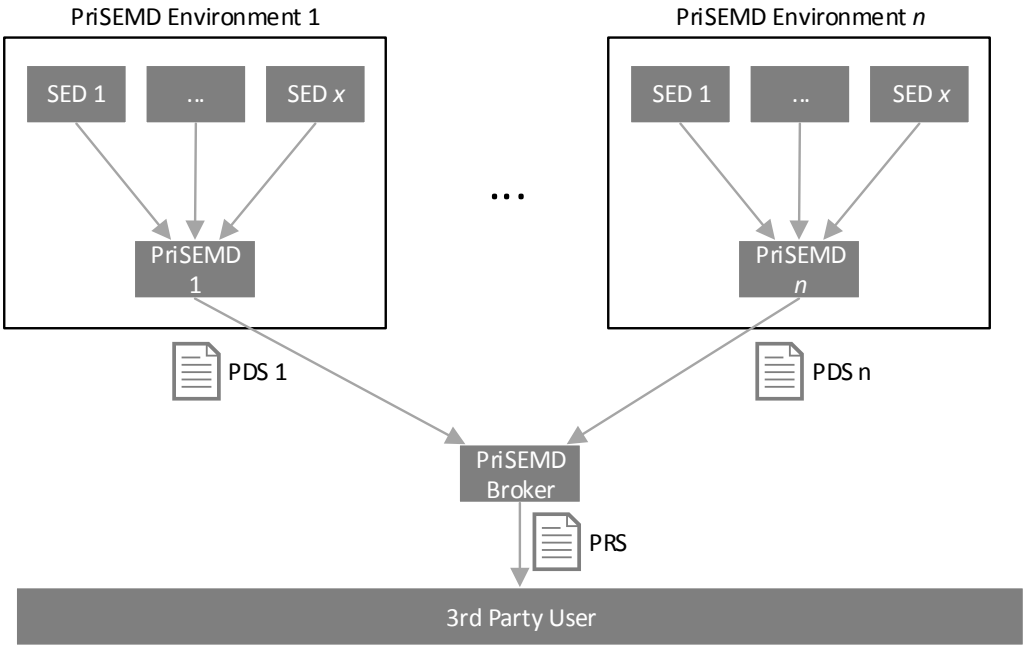



**Figure 21:** Anonymous Data Transmission Concept - General Overview

The measured data of PriSEMD which shall be forwarded to third parties is called **PriSEMD Data Set (PDS)**. Each PriSEMD device can submit a PDS to the PriSEMD Broker. All PDSs are received by the PriSEMD Broker which summarizes the PDSs to one result data set, called **PriSEMD Result Set (PRS)**. The Broker forwards the PRS to third party users. The structure of PDS is explained in detail followed by an examination of the PRS.

The PDS consists of information about when a specific device category is used, how old the user is and if the user is either male or female. Each PDS consists of PDS entries. A PDS entry is a 103 Bit vector which can be divided into four parts. Figure 22 shows which bit compares to which group of the PDS entry. The numbers in the gray boxes are showing the bit size of each group.

In the following listing each part of the PDS entry is described in detail:

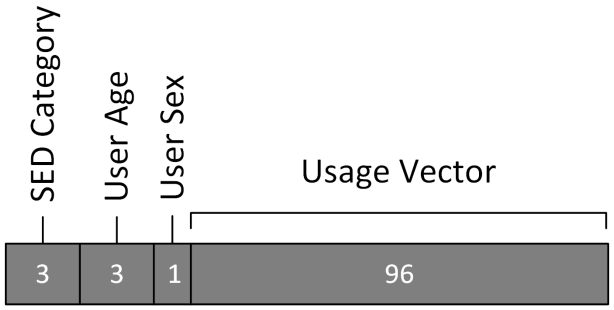



**Figure 22:** PDS Entry Structure

1. **SED Category Field**: The SED category field consists of three bits. Table 3 shows the mapping which bit combination belongs to which category of SED devices. The category *unknown device* can be used if the SED category cannot exactly be determined. All in all, there are three SED categories defined. If required, the SED category field has enough space to define four more SED categories.

| Number | SED Category | Binary | Number | SED Category | Binary |
|---|---|---|---|---|---|
| 0: | unknown device | 000 | 4: | not defined | 100 |
| 1: | STV | 001 | 5: | not defined | 101 |
| 2: | Sound System | 010 | 6: | not defined | 110 |
| 3: | Video Gaming Console | 011 | 7: | not defined | 111 |

**Table 3:** Definition of SED Category

2. **User Age Field**: The user age field consists of a three bit code representing the age span of the user. Table 4 shows an example of this code where seven different age spans are defined. A 38 year old user leads for example to a user age field entry of 100.

| Number | Age Span (in Year) | Binary | Number | Age Span (in Year) | Binary |
|---|---|---|---|---|---|
| 0: | 0 - 14 | 000 | 4: | 25 - 39 | 100 |
| 1: | 15 - 17 | 001 | 5: | 40 - 59 | 101 |
| 2: | 18 - 20 | 010 | 6: | 60 - 64 | 110 |
| 3: | 21 - 24 | 011 | 7: | 65 and older | 111 |

**Table 4:** Definition of User Age

3. **User Sex Field**: The gender of the user is stated using a single bit. A 0 is used for *male* and a 1 for *female*.

4. **Usage Vector Field**: The usage vector consists of 96 bit. Table 5 shows how the usage vector field is defined. The time span of one day is divided into 96 slices of each 15 minutes. The first slice is for example from *00:00 to 00:15*. The bit value consists of a 1 if the SED category is used in the given time interval, otherwise the field contains a 0.

| Time Interval: | 00:00 - 00:15 | 00:15 - 00:30 | ... | 23:45 - 00:00 |
|---|---|---|---|---|
| Usage: | 1 | 0 | ... | 0 |

**Table 5:** Definition of 96 Bit Usage Vector

The calculation of a PDS can be described formally: A PDS entry $pdse$ can be described as $pdse_{cas} = \{c, a, s, u\}$ whereas $c$ is the SED category in $C$, $a$ the user age in $A$ and $s$ the user sex in $S$. The usage vector $u$ can be described as $c = \{0, 0, 1, 0, \ldots, 0\} \in \{0, 1\}^{96}$. The size of $pdse$ is $size(pdse) = len(C) * len(A) * len(S) * len(U)$. A PDS $pds$ is defined as $pds = [pdse_{000}, pdse_{001}, pdse_{010}, \ldots, pdse_{071}, pdse_{100}, \ldots, pdse_{771}]$.

An example PDS entry could be $\{001|101|1|01100000\ldots0\}$. The SED category has the value 001 and says that the PDS entry belongs to a STV device. The user is between 40 and 59 years old (101) and female (1). The usage vector is not completely displayed here, but the fields shown are stating that an STV device was used during the intervals *00:15 - 00:30* and *00:30 - 00:45*.

The combination of SED category, user age and user sex within a PDS is unique. Assuming two users with the same age and sex are using the same SED category device. If this is the case, one usage vector exists for both persons. The uniqueness of the categories mentioned above leads to the fact that one PDS can consist of up to $2^7 = 128$ PDS entries.

The **PriSEMD Result Set (PRS)** and the PDS are similar structured, with one exception. The usage vector of a PRS entry does consist of chars not of bits. The PriSEMD Broker summarizes all received PDS entries to one PRS. For this purpose, the usage vectors of all PDS entries with the same SED category, user age and user sex field are summed up. Figure 23 shows a PRS entry constructed using three PDSs. The usage vectors of PDS1 and PDS2 are summed up, because the first three fields of both PDSs are the same. PDS3 is different and therefore not added.

The calculation of a PRS can be formally described as follows: A PRS entry $prse$ can be described as $prse_{cas} = \{c, a, s, u\}$ whereas $c$ is the SED category in $C$, $a$ the user age in $A$ and $s$ the user sex in $S$. The usage vector $u$ can be described as $c = \{x_0, x_1, \ldots, x_{95}\}$, whereas $x_n$ is the sum of all appropriate $pdse$ entries with the same $c$, $a$ and $s$. A PRS $prs$ is defined as $prs = [prse_{000}, prse_{001}, prse_{010}, \ldots, prse_{071}, prse_{100}, \ldots, prse_{771}]$.

$$
\begin{aligned}
PDS1 &: [\{'001','101','1','0|1|1|0|0|0|0|0|\ldots|0'\}] \\
PDS2 &: [\{'001','101','1','0|1|1|0|1|1|1|0|\ldots|0'\}] \\
PDS3 &: [\{'001','010','1','0|1|1|0|1|0|0|1|\ldots|0'\}]
\end{aligned}
\tag{1}
$$

$$
\begin{aligned}
PRS &: [\{'001','101','1','0|2|2|0|1|1|1|0|\ldots|0'\}, \\
&\quad \{'001','010','1','0|1|1|0|1|0|0|1|\ldots|0'\}]
\end{aligned}
$$

**Figure 23:** PRS - Example

For an attacker who wants to analyze the PRS it is difficult to achieve the information from which PriSEMD environment the data comes from. The summary is done by adding all PDS objects together. The result is called PriSEMD Result Set (PRS). The Broker forwards the PRSs to third party users.

Several assumptions have to be made. A malicious PriSEMD Broker is able to reconstruct which PSD comes from which PriSEMD environment. It is therefore assumed that the PriSEMD Broker is trustworthy and honest. Another assumption is that the data transmission cannot be manipulated and confidentiality is not violated. There has to be at least two PriSEMD environments for ensuring a correct working procedure of the PriSEMD Broker.

#### Example

An example of the concept previous introduced is shown in Figure 24. The example contains two PriSEMD environments where the left environment consists of a STV and a PriSEMD device, and the right environment of a STV, a video gaming console and the PriSEMD device.

The PDS object stated in Figure 25 is sent from PriSEMD environment 1 to the PriSEMD Broker. It contains two PDS entries which are both belonging to the STV device.

The PDS object of PriSEMD environment 2 is shown in Figure 26. It consists of three PDS entries, two belonging to the STV device, one to the video gaming console.

The PriSEMD Broker summarizes the PDS data objects and creates a PRS object. The PRS object created out of the PDS of environment 1 and 2 is shown in Figure 27. The first and the second entry of both PDSs are summarized by summing up the usage vector. $'0|0|0|1|1|0|0|0|\ldots|0' +' 0|1|1|1|1|1|0|0|\ldots|0' =' 0|1|1|2|2|1|0|0|\ldots|0'$. Each part of the vector is summed up with the equivalent part of the other usage vector. The same procedure is executed for the second usage vector of both PDSs. The third PDS entry of $pds_2$ has no other equivalent PDS entry and is therefore not modified within the PRS. Finally, the PRS object (shown in Figure 27) is forwarded to the third party users.

#### Size of PDS and PRS Data Objects

The maximum size of a PDS and PRS data object will be determined in this paragraph. As stated before, a PDS object consists of up to 128 PDS entries and each PDS entry has a size of 103 bit. The calculation shown in Figure 28 determines the maximum size of a PDS to 1.648Kbyte.

A PRS object consists also of up to 128 entries. The SED category, user age, and user sex field need 7 bits. The usage vector is a string with a size of 96 characters of which each has a size of 8 bit. As shown in Figure 29, the maximum size of a PRS data object is 12.4 Kbyte.

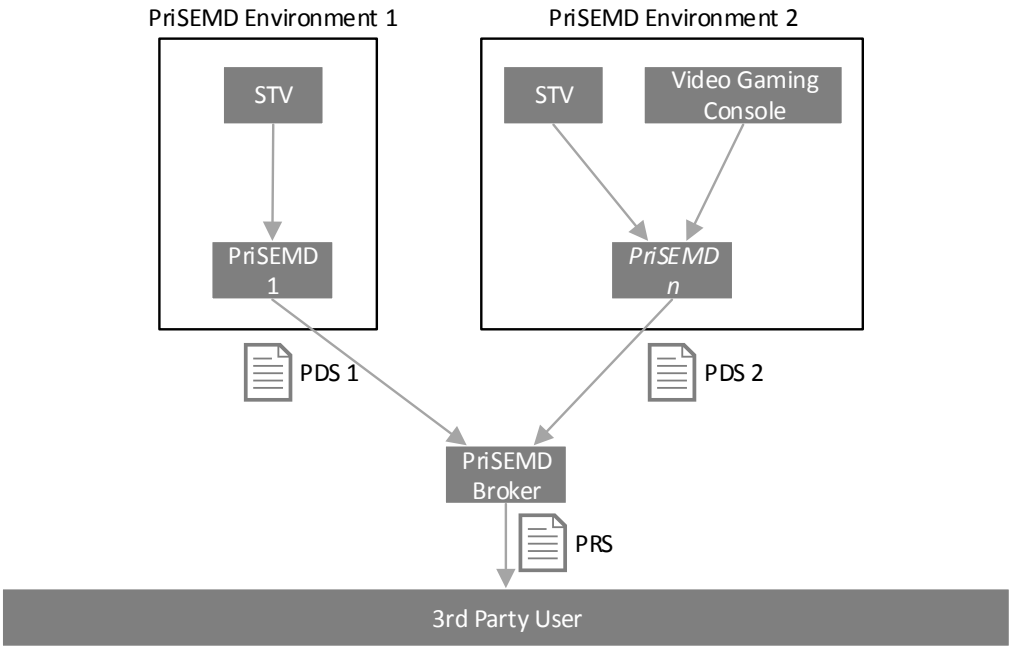



**Figure 24:** Anonymous Data Transmission Concept - Example

$$pds_1 = [\{'001','100','1','0|0|0|1|1|0|0|0|\ldots|0'\} \\ \{'001','010','1','0|0|0|1|1|0|0|0|\ldots|0'\}] \quad (2)$$

**Figure 25:** PDS PriSEMD Environment 1

$$pds_2 = [\{'001','100','1','0|1|1|1|1|1|0|0|\ldots|0' \\ \{'001','010','1','1|1|0|0|1|1|0|0|\ldots|0'\} \\ \{'011','010','1','1|0|0|0|0|0|0|0|\ldots|0'\}] \quad (3)$$

**Figure 26:** PDS PriSEMD Environment 2

$$prs = [\{'001','100','1','0|1|1|2|2|1|0|0|\ldots|0'\} \\ \{'001','010','1','1|1|0|1|2|1|0|0|\ldots|0'\} \\ \{'011','010','1','1|0|0|0|0|0|0|0|\ldots|0'\}] \quad (4)$$

**Figure 27:** PRS PriSEMD Broker

$$size(pds) = 128 * 103bit = 13184bit \\ = 1648byte \\ = 1.648Kbyte \quad (5)$$

**Figure 28:** PDS Data Object Size in Kbyte

$$
\begin{aligned}
size(prs) = 128 * (7bit + (96 * 8bit)) &= 99200bit \\
&= 12.400byte \\
&= 12.4Kbyte
\end{aligned} \tag{6}
$$

**Figure 29:** PRS Data Object Size in Kbyte

# 4 Prototype Implementation

The complete implementation effort done for this thesis is outlined in the following. The network infrastructure of the prototype is introduced in the first part of this Chapter 4.1. The network infrastructure is an environment in which the PriSEMD prototype is embedded in. Afterwards, each hardware component of the network infrastructure is explained in detail. The implemented software prototype is focused in Chapter 4.2.

## 4.1 Prototype Network Infrastructure

The network infrastructure in which the PriSEMD prototype is embedded in consists of three hardware components. As shown in Figure 30, there are a STV, a computing device and a router. The STV is the SED used in context of the prototype implementation and will be analyzed by PriSEMD. The computing device is the Linux based host of the PriSEMD software prototype and the router is used to obtain connection to the Internet. More details on the STV and the computing device are given in Chapter 4.1.1 and 4.1.2.

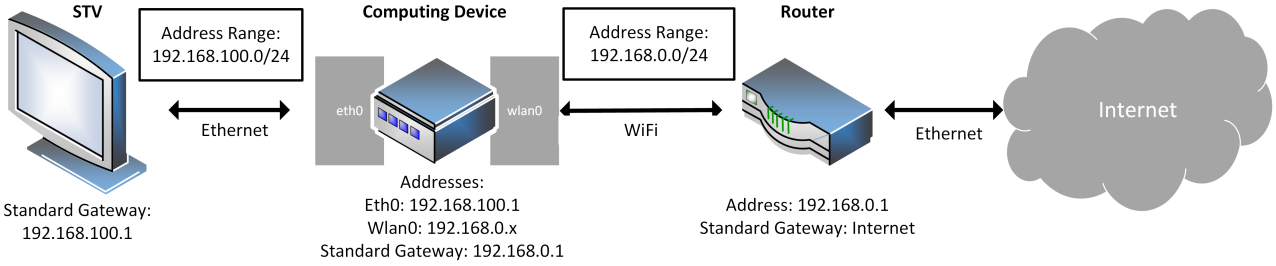



**Figure 30:** Prototype Network Infrastructure

The network architecture is based on the conceptual architecture from Chapter 3.1. The STV is connected to the computing device, the computing device to the router and the router is further connected to the Internet. The connection between the STV and the computing device, as well as the connection between the router and the Internet is implemented with cable based Ethernet. In contrast, the connection between computing device and router is realized using wireless Wi-Fi. The computing device network interface *eth0* is used for the Ethernet and the *wlan0* interface for Wi-Fi connection.

The connection methods mentioned above are used, because the computing device has only two network interfaces by default, an Ethernet port and a Wi-Fi option. The Ethernet port is used to connect to the STV, because it is the simplest way to do the connection. Using Wi-Fi for this connection part leads to the problem that a Wi-Fi hotspot has to be installed on the computing device which accompanies with a lot of implementation effort. An extension of further SEDs is still possible while using a switch between the SEDs and the computing device. However, a solution where all connections are realized using Ethernet is conceivable for later implementations, because Wi-Fi traffic can be monitored and eavesdropped. A cable based solution would avoid this risk.

The detailed network architecture is explained hereafter. One important aspect to mention is, that there are two different network sections. The first section is between STV and computing device's *eth0* interface, the second between the computing device's *wlan0* interface and the router.

The main reason why two different network areas are used is to record the network traffic of all available SEDs. Therefore, the SEDs' network traffic is routed to the computing device which is working as a proxy. Traffic arriving on the *eth0* interface is forwarded to the computing device's *wlan0* interface and vice versa. The computing device is able to trace outgoing (SED -> Internet) and incoming (SED <- Internet) network traffic of the SEDs.

The IP address range of the network between STV and computing device is from 192.168.100.2 to 192.168.100.255. The computing device has installed a DNS and DHCP server and the *eth0* interface has the address 192.168.100.1. Each device which is configured to receive network configurations from a DHCP server is assigned an IP address and a DNS server name automatically if it is connected to the *eth0* interface. Therefore, it is easy to connect other SEDs to the computing device and reduce the network configuration effort to a minimum. The standard gateway of the STV is set to the computing device's *eth0* interface.

The network between the computing device and the router is depending on the environment in which the PriSEMD prototype is implemented in. For testing purpose the following configurations are set up. The address range is from 192.168.1.2 to 192.168.1.255. The router has the fixed address 192.168.0.1 and runs a DHCP and DNS server. The computing device's *wlan0* interface is automatically assigned an address in the address range mentioned above. The standard gateway of the computing device is the router.

### 4.1.1 STV - Samsung UE40ES6300

The STV used in context of the prototype implementation is from 2012 and has the model number *UE40ES6300*. After stating a few things about the feature set of the test device, the STV is going to be analyzed in more detail.

The Samsung *UE40ES6300* is a typical representative of a STV. It comes with several hardware interfaces and software features. All features and interfaces described in Chapter 2.2 are as well available for the test device used. For that reason, the features are not repeated here again.

#### Measurable Interfaces on Test Device

As mentioned in Chapter 3.2, the three information sources infrared signals, network traffic and UPnP services are analyzed by PriSEMD. The Samsung test device has all three interfaces available. The STV has integrated an infrared sensor in order to communicate with a remote control, two network interfaces for both, Wi-Fi and Ethernet, and it supports UPnP. The appropriate remote control has the model number *AA59-00581A*. Performing a port scan on the STV shows that UPnP was supported by the STV. After describing the results of the port scan, details concerning the UPnP functionality are outlined.

**Port Scan**
A **port scan** is a network test where a client sends request messages to a host. The request messages are addressed to a range of ports in order to detect opened ports on a host [43, p. 229]. Usually, a port scan is used to detect network functionalities. It is executed by a software called **port scanner**. For port scanning the Samsung STV the common port scanner nmap[a] was used. Nmap was executed with the following command: *nmap -p 1-65535 -T4 -A -v 192.168.100.10*. The scan results are listed in the Table 6.

[a] http://nmap.org/

| Port | Name |
|---|---|
| 80/tcp | HTTP |
| 443/tcp | HTTPS |
| 3697/tcp | Nw-system: NavisWork License System |
| 4443/tcp | pharos |
| 6000/tcp | X11 |
| 7676/tcp | UPnP |
| 9090/tcp | Zeus admin |
| 9900/tcp | Iva |
| 55000/tcp | Remote control (can be used to send remote control signals) |
| 55001/tcp | ? |

**Table 6:** Port Scan on Test Device

Table 6 is divided into two columns. Column one shows the number of the opened port and the used transport layer protocol. Column two displays the name description returned by nmap. The results in Table 6 are showing that the STV supports UPnP on port 7676. Not focused in this work but also interesting to mention is that port 80 and 443 are opened. This is an indicator for a running web server on the STV. Interacting with the opened port 55000 allows network devices to send remote control signals to the STV in order to control it. The functionalities of the other opened ports 3697, 4443, 6000, 9090, 9900 and 550001 cannot exactly be determined and are therefore not considered in the following.

#### UPnP Service

As mentioned in the previous paragraph, the Samsung test device has an opened port for UPnP. All functionalities provided by this interface can be inspected with a UPnP explorer like gUPnP[6].

**gUPnP - UPnP Explorer Software**
The software gUPnP is a UPnP explorer which makes it possible to inspect all UPnP using devices and the accessible services in a network. After starting the software, gUPnP discovers all available UPnP devices in the local network. It is possible to request services and inspect the returning response messages [36].

[6] https://wiki.gnome.org/action/show/Projects/GUPnP?action=show&redirect=GUPnP#Download

GUPnP discovers all UPnP devices in the local network and provides an overview of available UPnP services. A few services of the STV's UPnP functionality are outlined in the following listing:

- **GetVolume**: The GetVolume UPnP service offers the opportunity to receive information about the current volume status of the test device. After sending a request to the service, the STV returns an integer value, representing the volume amount between 0 and 100.
- **SetVolume**: The SetVolume functionality gives a user the possibility to change the volume setting. An integer value representing the desired volume amount has to be provided in the request message. After sending the request message to the STV, the volume is changed. An appropriate response message confirms that the value change was successful or not.
- **CheckPIN**: The CheckPIN service allows a user to send a PIN Code to the STV. After sending a request message including the requested PIN message, the STV returns a message stating the PIN was either correct or invalid.
- **RunBrowser**: This service allows a remote user to start the STV's browser. A URL has to be provided in the request message. After receiving the message, the STV starts the Browser and calls the provided URL.

All in all, the test device has a huge amount of UPnP services available. A complete list of all services can be found in the appendix 23.

### 4.1.2 Computing Device - Cubietruck

The implemented prototype runs on the computing device called Cubietruck. Cubietruck is a single board computer(SBC) developed by Cubietech. It is the third version of the Cubieboard series and is very similar to the popular SBC Raspberry Pi. The Cubietruck is equipped with all necessary computer features like display, audio and USB ports. It has as well an Ethernet port and the ability to communicate with a wireless network using Wi-Fi. Furthermore, it has an integrated Infrared sensor [9]. The running operation system is called Linaro, a Linux Debian distribution optimized for the Cubietruck SBC.

## 4.2 Prototype Software Implementation

The prototype implementation is an attempt to implement the PriSEMD concept introduced in Chapter 3. The implementation focuses on the following aspects:

- showing that it is possible to measure SED interfaces(infrared, network, UPnP).
- showing that it is possible to extract user behavior related data out of the measured.
- presenting results in a GUI

Concerning the network architecture of Figure 30 the prototype implementation is located between the STV test device and the router. The prototype software is running on the Cubietruck.

An overview of the prototype implementation is given in Figure 31. Overall, there are six different components. The **Infrared**, **UPnP** and **Network Analyzer Agent** are raw data agents belonging to the *Collection Phase* of the PriSEMD concept. All three agents are collecting data and forwarding them to the **local database** where the data is saved. The **Data Mining Agent** is working on the *Processing and Analysis Phase* of the PriSEMD concept. The agent requests raw data from the local database, processes and extracts interesting information out of it. The **User Interface** consists of several visualizations showing the results of PriSEMD prototype measurement.

All implementation effort is done with the programming script language Python. The database is using MySQL and the user interface is based on a webpage accessible through an Apache Web Server. Each software agent is explained successively in the following subsections. Finally, the user interface is illustrated with a few screenshots.

### 4.2.1 Infrared Agent

The **Infrared Agent** is a software program receiving infrared signals and forwarding them to the local database. The implementation of the Infrared Agent and the problems faced in the implementation are outlined hereafter.

The Infrared Agent is a passive raw data agent working on the *Collection Phase* (3.1.1) of the PriSEMD concept. General aspects concerning infrared data gathering are mentioned in Chapter 3.2.3. As shown in Figure 32, the computing device is established near to the STV test device. The close proximity of both devices allowing the Cubietruck to receive the STV's remote control infrared signals.

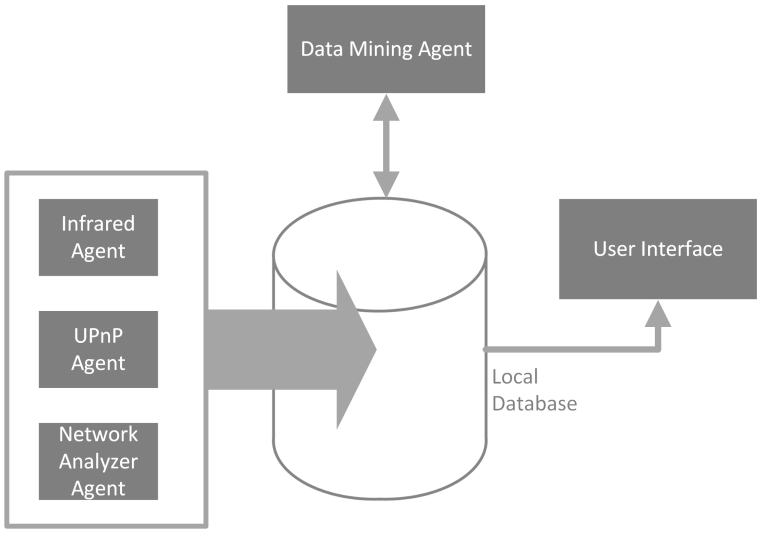



**Figure 31:** TV Analyzer Implementation Overview

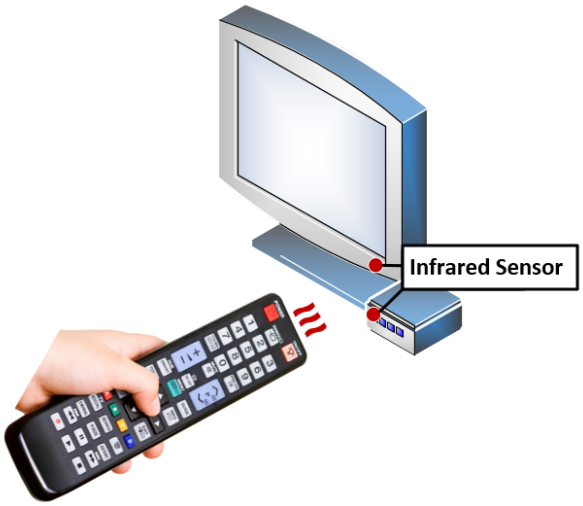



**Figure 32:** Infrared Signal Capturing[7]

The implementation of the Infrared Agent uses the Linux Infrared Remote Control (LIRC) tools to interact with the Cubietruck's infrared sensor. LIRC provides several programs allowing the Infrared Agent to request signals recorded by the infrared sensor. The data collected by LIRC is piped into the Infrared Agent and will be afterwards forwarded to the local database. In the following, it will be explained how the LIRC tools work and how the Infrared Agent Script is implemented.

#### LIRC

LIRC is an open source collection of tools that can be used to interact with an infrared sensor. The usage of LIRC requires installing specific drivers and setting up a few configurations. The needed drivers are depending on the infrared sensor model and the used Linux distribution. A step by step guide how to setup LIRC on the Cubietruck can be found under [55].

After installing LIRC and all necessary drivers, one more configuration has to be done. Due to a lack of infrared signal standardization, each manufacturer follows its own way of defining IR signals. A **remote control configuration file** is needed where each IR signal is mapped to a defined button on the remote control. The power button has for example the IR signal 0x1020045345029. Every time LIRC receives the IR signal 0x1020045345029 it will be mapped to the pressed power button.

[7] source of remote control image: [57]

A lot of configuration files can be found on the LIRC web page[8]. However, this list is outdated, most files are from 2008 to 2010. In the previous stated archive no fitting configuration file is available for the remote control *AA59-00581A*. Therefore, a new configuration file has to be created.

The *irrecord* program, distributed with the LIRC package, can be used for this purpose. With this program, it is possible to read in each key signal of the remote control and assign an appropriate button name to it. The button name can be chosen from a list of key names provided by *irrecord*. After reading in each key, *irrecord* returns the needed configuration file. This configuration file contains pairs of IR signal codes and human readable names. Furthermore, it consists of details about the vendor and the model number of the remote control. The vendor and model number information can either be filled in manually, by editing the configuration file, or it can be stated during the *irrecord* recording process [2, 58].

Before the created remote control configuration file can be used, the values in the file has to be edited manually. For each IR signal a specific offset value has to be subtracted from the IR signal code. Afterwards the configuration file is accepted by LIRC and the signal capturing can be started.

The LIRC program *irw* can be used to get all incoming IR signals from the sensor. This program uses the configuration file created with *irrecord*. *irw* receives the incoming signal information from the IR sensor and maps the signal code to the button stated in the configuration file. For each received IR signal *irw* outputs information about the name of the pressed button, the IR signal hex code and the name of the remote control directly on the console.

#### Infrared Agent Implementation

The Infrared Agent uses *irw* to obtain infrared signal data. For this purpose, the console output created by *irw* is piped into the Infrared Agent. The output consists of one line of data for each infrared signal recorded. Each output line is processed in the implementation. A timestamp is added to the signal information due to the fact that the moment of receiving is very important. The information of *irw* and the timestamp are forwarded to the local database.

Table 7 shows an example. The first entry is an output line of *irw*. The second entry shows the data base entry belonging to the *irw* output line.

| | |
|---|---|
| irw output: | 0000000000001002 00 KEY_POWER Samsung |
| forwarded to DB: | Timestamp: 2014-09-07 16:35:58<br>Button Name: KEY_POWER<br>Manufacturer Name: Samsung<br>Signal Code(hex): 0000000000001002 |

**Table 7:** Infrared Agent Example

### 4.2.2 Network Analyzer Agent

The **Network Analyzer Agent** is a software program that reads and records the network traffic between SEDs and Internet. All network packets sent or received by the SEDs are analyzed in detail and relevant information being extracted by the agent.

The Network Analyzer Agent is a passive raw data agent working on the *Collection Phase* (3.1.1) of the PriSEMD concept. General aspects concerning network traffic recording are mentioned in Chapter 3.2.1.

The implemented agent records two different types of network packets. These packets can be categorized by the used application layer protocol - HTTP and SSDP. Figure 33 illustrates how these protocols are constructed. The package Scapy is used to record and inspect network packets. The following paragraph introduces the functionalities provided by Scapy.

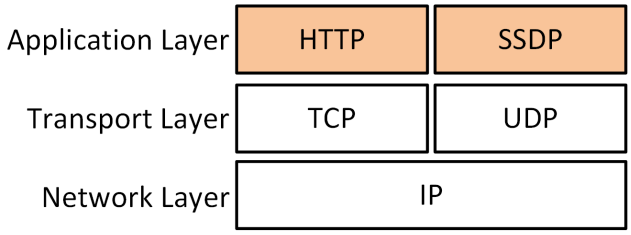



**Figure 33:** Rule Engine Example[9]

[8] http://lirc.sourceforge.net/remotes/

[9] based on: [50]

#### Scapy

The common used python package Scapy[10] is a tool set which offers the opportunity to manipulate and sniff network packets. Only the sniffing functionality is used in the Network Analyzer Agent.

In the context of Scapy, network packets are represented as a layered structure. The layered structure can be compared with the OSI model levels. A HTTP packet is for example represented as HTTP(TCP(IP(...))) [54, p. 136]. Two options have to be configured before it is possible to run Scapy:

1. The network interface on which Scapy shall sniff on has to be defined. As shown in Figure 30, the STV is connected to the *eth0* interface of the TV Analyzer. For that reason, Scapy is initialized with the *eth0* interface. Sniffing on this interface makes it possible to only record traffic belonging to the connected SEDs.

2. Scapy needs an appropriate filter setting. This filter defines which network packets are considered by Scapy. All packets that are excluded by the filter are dropped by Scapy. In view of the fact that the Network Analyzer Agent shall record HTTP and SSDP messages the desired filter setting looks like *IP AND (TCP OR UDP) AND (HTTP OR SSDP)*. However, Scapy can only filter protocols up to the transport layer. Application Layer protocols are not considered by Scapy. Therefore, the Scapy filter is defined with the term *IP AND (TCP|UDP)*. The missing AND connection from the desired filter mentioned above has to be implemented in the Network Analyzing script itself. The implementation is described in the next section.

#### Network Analyzer Script Implementation

The Network Analyzer Agent is a python script which uses the packet Scapy to analyze network packets. For each packet corresponding with the provided filter setting the agent jumps in a specific function of the script. The missing filter term is implemented in this function. The filter part *AND (HTTP or SSDP)* is implemented using several if clauses checking if the packet is either HTTP or SSDP. Each packet using neither HTTP nor SSDP is dropped by the script.

The agent is able to differentiate six different packet classes. An overview of all possibilities can be found in Table 8. The first column displays the name of the recorded packet class category. The second column shows the used transport and application layer protocol, and the last column states how a packet is characterized. The characterization is done by checking the packet content and finding appropriate key words. Three entries of Table 8 are described as examples:

The first example is a packet of the category **HTTP Request**. A **HTTP Request** is found if the recorded packet uses the protocols TCP and HTTP. Furthermore, either the keyword *GET* or *POST* has to be stated in the header of the HTTP protocol section.

The second example describes how a **SSDP - alive** packet can be found. The SSDP message structure is similar to the structure of HTTP messages. For that reason, a recorded SSDP packet can be found while searching for a packet using UDP and HTTP protocols. A SSDP-alive packet has to have further the header *ssdp:alive* set.

The last example states how a **HTTP Response - HbbTV** packet can be found. Besides using the protocols TCP and HTTP, the recorded packet has to contain the keyword *response* in the header section of the HTTP protocol section and the keyword *hbbtv* in either the header or body part of the HTTP protocol section.

| Packet Name | Protocol Characteristic | Header Characteristic |
|---|---|---|
| HTTP Request | TCP AND HTTP | HTTP Header: 'GET'/'POST' is set |
| HTTP Response | TCP AND HTTP | HTTP Header: 'response' is set |
| HTTP Request - HbbTV | TCP AND HTTP | HTTP Header: 'GET'/'POST' is set AND HTTP Header/-Payload contains 'hbbtv' |
| HTTP Response - HbbTV | TCP AND HTTP | HTTP Header: 'response' is set AND HTTP Header/Payload contains 'hbbtv' |
| SSDP - alive | UDP AND HTTP | SSDP Header: 'ssdp:alive' is set |
| SSDP - byebye | UDP AND HTTP | SSDP Header: 'ssdp:byebye' is set |

**Table 8:** Recorded Packets and Characteristics of Network Analyzer Agent

### 4.2.3 UPnP Agent

The **UPnP Agent** is a script that request UPnP services of SEDs and forwards the requested information to the local database. The UPnP Agent of the PriSEMD prototype requests data from the STV test device. In this section it is introduced how UPnP services can be programmatically requested and which information can be requested in detail.

The UPnP Agent is an active raw data agent working on the *Collection Phase* (3.1.1) of the PriSEMD concept. General aspects concerning UPnP data gathering are mentioned in Chapter 3.2.2.

[10] http://www.secdev.org/projects/scapy/

#### Programmatically Communicate with UPnP Services

An approach to request data from UPnP services is introduced in this paragraph. The requesting process consists of sending self-constructed network packages to the desired UPnP services. These packages are constructed by the UPnP Agent and sent to the SED's UPnP interface.

Information about the structure of a UPnP request packet can be explored with a UPnP browser tool. Using such a tool, it is possible to request desired UPnP services of available devices. After recording the network traffic between UPnP browser tool and SED, the requested message has to be analyzed. The knowledge about the request message structure can be used to rebuild the message in the UPnP Agent program.

As mentioned in Chapter 3.2.2, the UPnP requesting mechanism is based on SOAP. Before it is possible to send self-constructed request packets to the SEDs UPnP interface, a SOAP client functionality has to be implemented in the UPnP Agent script.

The previous stated UPnP browser tool gUPnP was used in order to analyze UPnP request messages. Further details about gUPnP are stated in 4.1.1. The program wireshark[11] can be used to analyze network traffic and inspect details of network packets. Two python packages are used for the implementation. The *soappy*[12] package is used to send self-constructed SOAP messages to a SOAP using host. The package *Beautiful Soup*[13] is used for parsing HTML content.

The solution introduced here for requesting UPnP services is only one of many possibilities. Another way to implement the functionality can be achieved while implementing a UPnP client functionality in the UPnP Agent script. Therefore, python packages like *coherence*[14] or *miniupnp*[15] can be used. Though, there are several problems coming along with the usage of these packages. Either they are insufficient documented or too complicated in use. Mostly essential examples are missing which represents another barrier for the usage of these packages.

#### UPnP Agent Script Implementation

The UPnP Agent is a program that polls a list of specific UPnP services in a defined time interval. The time interval can be easily changed, for testing purpose the interval is set to *five seconds*. Each interval, the UPnP Agent script requests the defined UPnP services.

The requested UPnP services are listed in Table 9. All in all, one polling cycle requests information that can be divided into four different categories - TV Information, Browser Information, Channel Information and Information about External Sources. The listing below describes which information is being requested and which service has to be polled in detail. Further, the name of the requested services is stated in context of each category.

1. **Channel Information**: Data associated with the current running channel is gathered in this part. These include information about the channel name, the program title, the volume status of the TV and the status of the mute functionality. The requested services of the STVs UPnP interface are *GetCurrentContentRecognition*, *GetCurrentMainTVChannel*, *GetVolume* and *GetMute*.

2. **Information about External Sources**: Here, data concerning external signal sources are requested. An external signal source is an interface that can be used to interact with external devices. *HDMI* or *SCART* are e.g. external signal sources. An external device is a gadget connected to the STV offering other functionalities that can be used in combination with the STV. Video gaming consoles or disk players are for example external devices. The polled service of the UPnP interface is *GetCurrentExternalSource*. The service provides a different value concerning which external source is currently activated. The service mentioned above returns for example *HDMI* if the STV is currently interacting with an external device via the HDMI port. Furthermore, it is possible to check if the test device is in the TV-Mode (return value: TV) or if the browser is currently running.

3. **Browser Information**: Information about STVs integrated web browser are requested in this part. The web browser of a STV can be compared to a desktop based browser. The polled services of the UPnP interface are *GetCurrentBrowserMode* and *GetCurrentBrowserURL*. The requested information is about browser mode and the current requested browser URL. The browser-mode information describes if the browser is currently active or inactive. The current requested browser URL provides the URL of the actually present web page on the browser.

4. **TV Information**: Here, data concerning the STVs display setting are polled. These include information about brightness, sharpness, contrast and color-temperature of the STV display. The polled services of the UPnP interface are *GetBrightness*, *GetSharpness*, *GetContrast* and *GetColorTemperature*. All services are providing integer values between 0 and 100 describing the current setting.

[11] `https://www.wireshark.org/`
[12] `https://pypi.python.org/pypi/SOAPpy`
[13] `http://www.crummy.com/software/BeautifulSoup/`
[14] `http://coherence.beebits.net/`
[15] `http://miniupnp.free.fr/`

The data is forwarded to the local database after polling the data from the specific interface(s). However, the data is only saved, when it differs from the entry before. The data from the TV information part is e.g. only saved, when one value is changed during the last interval. This fact is being considered, in order to avoid multiple entries with same content and reduce the amount of saved data in the end. A summary of the services used by the UPnP Agent are shown in Table 9.

| Category | UPnP Service Name | Description |
|---|---|---|
| TV Information | GetBrightness | Brightness setting of the display |
| TV Information | GetColorTemperature | Color Temperature setting of the display |
| TV Information | GetContrast | Contrast setting of the display |
| TV Information | GetSharpness | Sharpness setting of the display |
| Information about External Sources | GetCurrentExternalSource | Used external source, TV or Browser |
| Browser Information | GetCurrentBrowserMode | Browser status: active or inactive |
| Browser Information | GetCurrentBrowserURL | Current accessed URL |
| Channel Information | GetCurrentContentRecognition | Name of current running channel and program |
| Channel Information | GetCurrentMainTVChannel | Current channel number |
| Channel Information | GetVolume | Volume amount |
| Channel Information | GetMute | Mute setting: on or off |

**Table 9:** Requested UPnP Information of STV Test Device

### 4.2.4 Data Mining Agent

The **Data Mining Agent** is the most complex agent built for PriSEMD. Figure 34 shows an overview of the Data Mining Agent and where it is located in the overall structure of PriSEMD. It is a software working on the *Processing Phase* (3.1.2) and *Analyzing Phase* (3.1.3) of the PriSEMD concept. The Data Mining Agent combines the data collected by the Network Analyzing, Infrared and UPnP Agent and creates a consistent data structure.

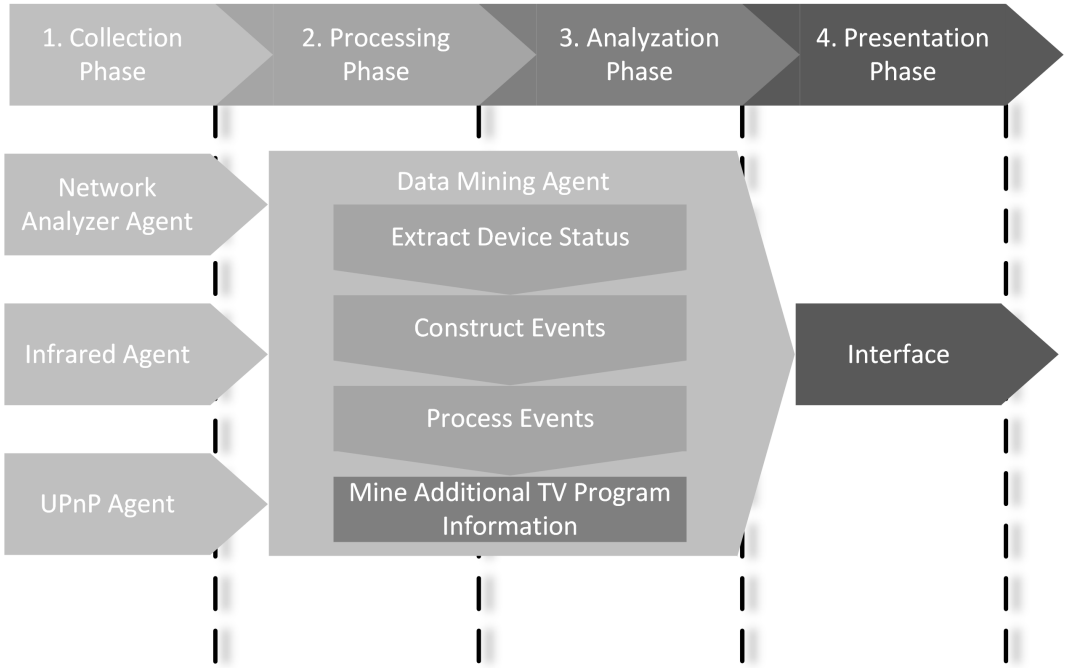



**Figure 34:** Data Mining Agent Overview

Hence, the functionality of this agent can be divided up into four different parts executed step by step. An overview about these four parts gives the listing below:

1. **Extract Device Status**: Step one is a preprocessing phase where information about the on/off status of a SED is extracted out of the provided raw data.

2. **Construct Events**: Step two is responsible for constructing events and building the consistent event structure. A detailed explanation of the event structure is provided in Chapter 3.1.2. This step is located in the *Processing Phase*.

3. **Process Events**: Step three is a post processing phase where multiple events are comprehended in order to shrink the total amount of events.

4. **Mine Additional TV Program Information**: Additional information about the TV program are requested from a website and connected to appropriate entries of the event structure. The fourth step is located in the *Analysis Phase* of the PriSEMD concept.

Each step of the Data Mining Agent will be described in the following sections.

#### 1. Extract Device Status

In the first step of the Data Mining Agent, information about the device status is extracted from the raw data. The analyzed raw data consists of the SSDP messages sent out from the SED and recorded by the Network Analyzing Agent. The goal of this step is to find out when the SED is in one of three states belonging to the power status of the device. The three states are *on*, *running* and *off*. Figure 35 shows the relationship between the states and possible state transitions.

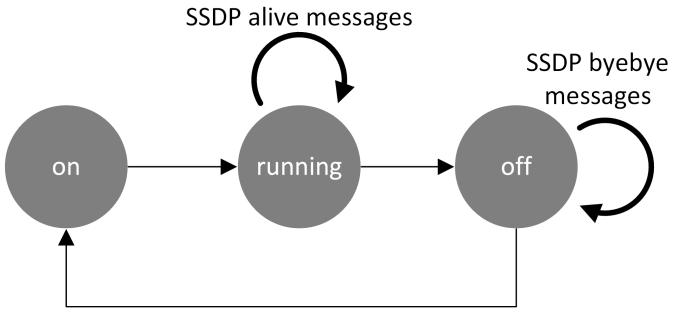



**Figure 35:** Device State Diagram

The following paragraph describes how to detect the current state of a SED. Relevant SSDP messages contain either the header field *ssdp:alive* or *ssdp:byebye*. These header fields are important for the analysis of the states mentioned above. Further details of the SSDP message structure can be found in Chapter 3.2.1. Receiving one of these messages leads to the finding that the device is either available or not. Two successively recorded packets have to be analyzed in order to find information about the previous shown three states. Table 10 shows which combination of SSDP messages leads to which state information.

| State | First SSDP Message Header | Second SSDP Message Header |
|---|---|---|
| on | ssdp:byebye | ssdp:alive |
| running | ssdp:alive | ssdp:alive |
| off | ssdp:alive | ssdp:byebye |

**Table 10:** SSDP Message Pairs and Device States

The first row of Table 10 shows which packets are required to detect that the device is turned on. Here, the transition from *ssdp:byebye* to *ssdp:alive* is searched. In other words, two packets have to be detected in sequence, first a packet with the header field *ssdp:byebye*, second with *ssdp:alive*. The detection that a device is running can be made while searching sequences of *ssdp:alive* messages. To detect the turned off status of a SED, the combination *ssdp:alive* and *ssdp:byebye* has to be searched.

Due to the fact that SSDP is sent using UDP, it is possible that a message is not received by the Network Analyzer Agent.

- **Case 1 - Device was turned on but no *ssdp:alive* message detected**: As a result of the fact that *ssdp:alive* messages are sent in regular intervals while the device is running, detecting the *on* state is delayed by one interval.

- **Case 2 - Device was turned off but no *ssdp:byebye* message detected**: Another possibility to detect that the device is turned off is for example to analyze if no *ssdp:alive* messages are received for a certain period of time. If there are no *ssdp:alive* messages received it can be assumed that the device was turned off.

In order to reduce the risk of packet loss, the test device sends each message multiple times directly one behind another.

For each found device status an entry in the local database is created. These entries are used in the second step of the Data Mining Agent in order to create the event structure. The here shown concept is a general analyzing strategy that can be applied to all SSDP using devices. In the context of the prototype implementation, only the STV test device is analyzed.

**2. Construct Events**

The event structure is created in the second step of the Data Mining Agent. The Event Processing Unit (EPU) is an implementation of the concept described in Chapter 3.1.2. As displayed in Figure 7 the EPU uses the raw data and a specified rule set in order to create events. The following is structured in three parts explaining the implementation of events, rules and the EPU.

**2a. Prototype Event Structure**

In the description of the Processing Phase (Chapter 3.1.2), an event consists of three parts - *Event Description*, *Event Level* and *Event Timestamp*. Within the prototype implementation two parts are added - an *ID* and an *Upper Level ID*. The *ID* is a unique identifier for the event. The *Upper Level ID* is used to connect an event to an upper level event in order to create the level based event structure. An event level *x* has an *Upper Level ID* to an event with the level x-1. Due to the fact that level 1 events do not belong to an upper level event, they do not need an *Upper Level ID*. Figure 36 shows an example event structure. The example consists of six different events from level 1 to 3.

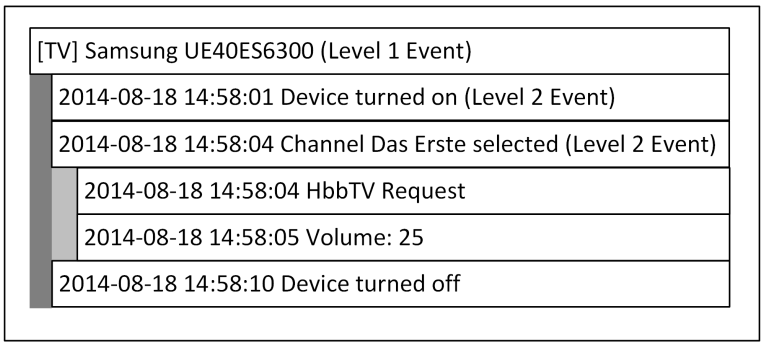



**Figure 36:** Example Event Structure

The data base used for the example shown in Figure 36 is displayed in Table 11.

| ID | Description | Event Level | Upper Level ID |
|---|---|---|---|
| 1 | 2014-08-17 14:58:01 [TV] Samsung UE40ES6300 | 1 | - |
| 2 | 2014-08-18 14:58:01 Device turned on | 2 | 1 |
| 3 | 2014-08-18 14:58:04 Channel *Das Erste* selected | 2 | 1 |
| 4 | 2014-08-18 14:58:04 HbbTV Request | 3 | 3 |
| 5 | 2014-08-18 14:58:05 Volume: 25 | 3 | 3 |
| 6 | 2014-08-18 14:58:10 Device turned off | 2 | 1 |

**Table 11:** Example Event Structure - Data Base

**2b. Prototype Rule Set**

The assignment, which raw data leads to which specific event is defined by a rule. Each rule is a kind of blueprint for a class of events. That means, one rule can be the base of one or more events with similar attributes. A collection of rules is called **rule set**.

Each rule is described by the Rule Description Language (RDL) specially designed for that purpose. The RDL is flexibly designed in order to make an integration of new rules as easy as possible. Table 12 shows a Backus-Naur notation defining the content of each rule field. While inspecting the table, it is possible to get a clue of which value each field has to contain. Line (1) to line (11) are describing help variables and Line (13) to line (21) the rule fields.

Three terms have to be defined before outlining the exact structure of implemented rules. An **expression** is an equation consisting of two parts, the database table column name on the right side and the value on the left side. The expression *device_friendly_name = [TV]UE40ES6300]* consists for example of the table column name *device_friendly_name* and the appropriate value *[TV]UE40ES6300*.

| | | | |
|---|---|---|---|
| ⟨digitExceptZero⟩ | ⊨ | 1 . . . 9 | (7) |
| ⟨digit⟩ | ⊨ | 0 \| ⟨digitExceptZero⟩ | (8) |
| ⟨positiveNumber⟩ | ⊨ | ⟨digitExceptZero⟩ ⟨positiveNumber⟩ \| ⟨digit⟩ | (9) |
| ⟨letter⟩ | ⊨ | A . . . Z \| a . . . z | (10) |
| ⟨specialCharacter⟩ | ⊨ | _ | (11) |
| ⟨character⟩ | ⊨ | ⟨digit⟩ \| ⟨letter⟩ \| ⟨specialCharacter⟩ | (12) |
| ⟨word⟩ | ⊨ | ⟨character⟩⟨word⟩ \| ⟨character⟩ | (13) |
| ⟨variable⟩ | ⊨ | <⟨word⟩> | (14) |
| ⟨eventText⟩ | ⊨ | ⟨word⟩ ⟨eventText⟩ \| ⟨variable⟩ ⟨eventText⟩ \| ⟨word⟩ \| ⟨variable⟩ | (15) |
| ⟨expression⟩ | ⊨ | ⟨word⟩=⟨variable⟩ \| ⟨word⟩=⟨word⟩ | (16) |
| ⟨expressionSequence⟩ | ⊨ | ⟨expression⟩,⟨expression⟩ \| ⟨expression⟩ | (17) |
| | | | (18) |
| ⟨id⟩ | ⊨ | ⟨positiveNumber⟩ | (19) |
| ⟨requirementTable⟩ | ⊨ | ⟨word⟩ | (20) |
| ⟨requirementValue⟩ | ⊨ | ⟨expressionSequence⟩ \| ⟨eventText⟩ \| any | (21) |
| ⟨sourceTable⟩ | ⊨ | ⟨word⟩ | (22) |
| ⟨sourceValue⟩ | ⊨ | ⟨expressionSequence⟩ | (23) |
| ⟨eventLevel⟩ | ⊨ | ⟨positiveNumber⟩ | (24) |
| ⟨eventDescription⟩ | ⊨ | ⟨eventText⟩ | (25) |
| ⟨eventDescription⟩ | ⊨ | ⟨word⟩ | (26) |
| | | | (27) |

**Table 12:** Rule Description Language Definition

An **expression sequence** is a list of expressions separated by a „ , “ character. *Status=device turned off, event_date=2014-02-30 14:11:38* is for example an expression sequence composed of two expressions.

A **variable** is a placeholder marked with a less „<“ and greater than „>“ character. The *<date>* variable is e.g. a placeholder for an entry like *28.05.2014 11:30:03*

Table 13 illustrates a rule example.

| Rule Field | Example Value |
|---|---|
| id | 10 |
| requirement_table | mined_data_events_level_2 |
| requirement_value | Channel: <channel-Name> |
| source_table | raw_data_upnp_channel |
| source_value | program_title=<programName>, channel_name=<channelName>, occurrence=<date> |
| event_level | 3 |
| event_description | <date> Program watched: <channelName> - <programName> |
| device | [TV] UE40ES6300 |

**Table 13:** Example Rule

Based on the description in Chapter 3.1.2, the prototype rule structure differs in three points from the concept. First, an additional id field is added to the prototype rule. Second, the requirement and source fields are both split up into *requirement/source_table* and *requirement/source_value* fields. Third, the timestamp field is integrated in *the event_description field*. Each *event_description* begins with a timestamp and ends with an informative text.

In combination with describing the example rule shown in Table 13 each rule field is explained in the listing hereafter. The complete rule set used within the prototype implementation can be found in Table 24 at the end of this thesis.

- **id**: The id field contains a unique number identifying each rule. The rule shown in Table 13 has the id value 10.

- **requirement_table**: The requirement_table field contains the name of a database table where the requirement information is stored.
- **requirement_value**: The requirement_value field describes where the relevant requirement information can be found in more detail. It describes which data of the requirement_table can be used for the requirement information. The requirement_value contains either an event description text, the static value *any* or an expression/ expression sequence. The example rule shown in Table 13 has the event description text *Channel: <channel-Name>* as requirement value. The requirement information can therefore be found in the database table *mined_data_events_level_2* and consists of all entries with event description text *Channel: <channel-Name>*. The variable *<channel-Name>* is replaced by the EPU while producing the events with a real channel name.
- **source_table**: The source_table field contains the name of a database table where the source information is stored.
- **source_value**: The source_value field contains an expression or an expression sequence. The combination of source_table and source_value can be used to find the source information in the same way as the requirement information can be found by requirement_table and requirement_value. The source information is used by the EPU in order to replace the variables in the event description. The example rule has an expression sequence consisting of three expressions.
- **event_level**: The event_level field contains a number describing the level of the produced event.
- **event_description**: The event_description field contains a string. This string is a template for the event description text. The template consists of variables replaced by EPU while producing the events. The rule shown in Table 13 has an event_description containing three variables. A later produced event description text could for example be read as follows: *2014-08-10 19:12:54 Program watched: RTL - News*.
- **device**: The device field contains an identifier of the device, on which the rule can be successfully run. The example rule in Table 13 can be executed in combination with the test device [TV] UE40ES6300.

An example, how a rule is used to calculate events is explained in the following section.

#### 2c. EPU Procedure

In this section it is explained how events are created by the EPU. The general EPU working procedure consists of going through the complete rule set step by step. Events are calculated within the definition of each rule and are produced from lower to higher level. In the prototype implementation events level 1 are hardcoded. Hence, level 2 is the level where the event production starts. After producing all level 2 events, events level 3 are constructed. The three steps shown below are performed for each rule of the provided rule set:

1. **part - Finding Event Source Information**
2. **part - Event Construction**
3. **part - Requirement Check**

In the following, each step is explained in detail and in combination with an illustrated example.

Source information for each event are requested from the local database in the first step of the EPU procedure. The *source_table* and *source_value* fields of the current rule describe where the source information can be found. The *source_table* field contains the table name and the *source_value* more details about which entries of the specific table are used for the source information. For each entry corresponding to the *source_table* and *source_value* fields an event is going to be constructed.

Figure 37 is showing an illustration of this example. The current rule is shown on the left side of the figure. As stated in the EPU procedure description above, the first step consists of requesting the source information. Therefore, a SQL statement is constructed and finally submitted to the database. The SQL statement is built using the *source_table* and the *source_value* field of the current rule. As shown in the rule description, the table name is *mined_data_device_on_off_status*. The *source_value* of the rule consists of three entries. *device_friendly_name=[TV]UE40ES6300* and *status=device turned on* are used for the WHERE part of the SQL statement. They are connected with an AND operator by default. The third entry of this field *event_date=<date>* is not considered for the WHERE part, because the right side of the expression consists of a variable. After executing the constructed SQL statement the query shown below is delivered by the local database. The variable *<date>* is connected to the date column of the query.

In the second step of the EPU procedure, the event is constructed using the source information requested in the previous step. As mentioned before, each event consists of four different fields which have to be filled with information.

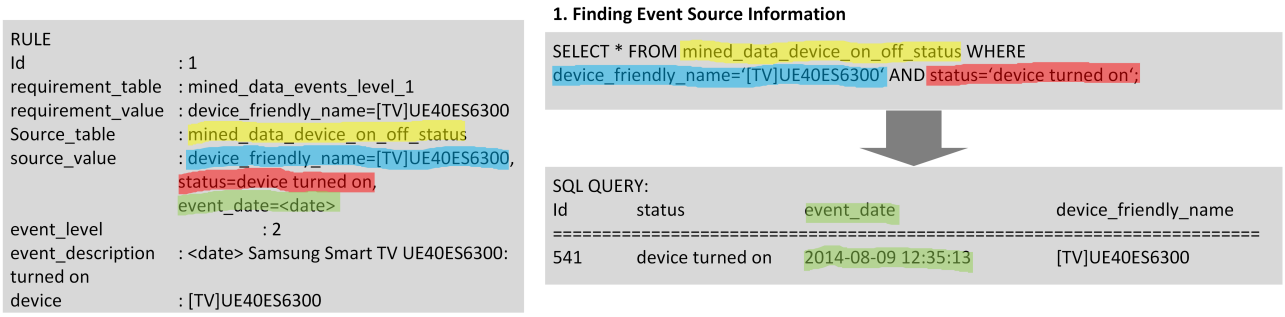



**Figure 37:** Rule Engine Example - Part 1: Finding Event Source Information

The *ID* field is managed by the database and not considered by the EPU. The *Description* field is filled with the text stated under the *event_description* of the current rule. The event description text usually consists of one or more variables. Each variable is replaced with data from the source information. The variable assignment in the *source_value* field of the rule tells the EPU which content has to be used for a specific variable. In the end, the variable is replaced with content of a specific column. The *Event Level* is set to the number stated in the current rules *event_level* field. The *Upper Level ID* is the next lower level. In other words, the *Event Level* subtracted with 1.

The event construction part of the example is shown in Figure 38. Assuming the query executed in step one consists of 34 entries, the same number of events are produced in step two - for each query entry one event. The event description is filled in with the template text stated in the rules *event_description* field. Due to the fact that the template text consists of the variable *<date>*, this part has to be replaced with content. The appropriate content which has to be inserted here is stated in the third entry of the *source_value* field. Text *event_date=<date>* tells the EPU that the variable *<date>* has to be replaced with the content stated in the column *event_date* of the query from step one. In the event description the *<date>* is replaced with the appropriate date of the query. The event level is stated under the *event_level* field of the rule.

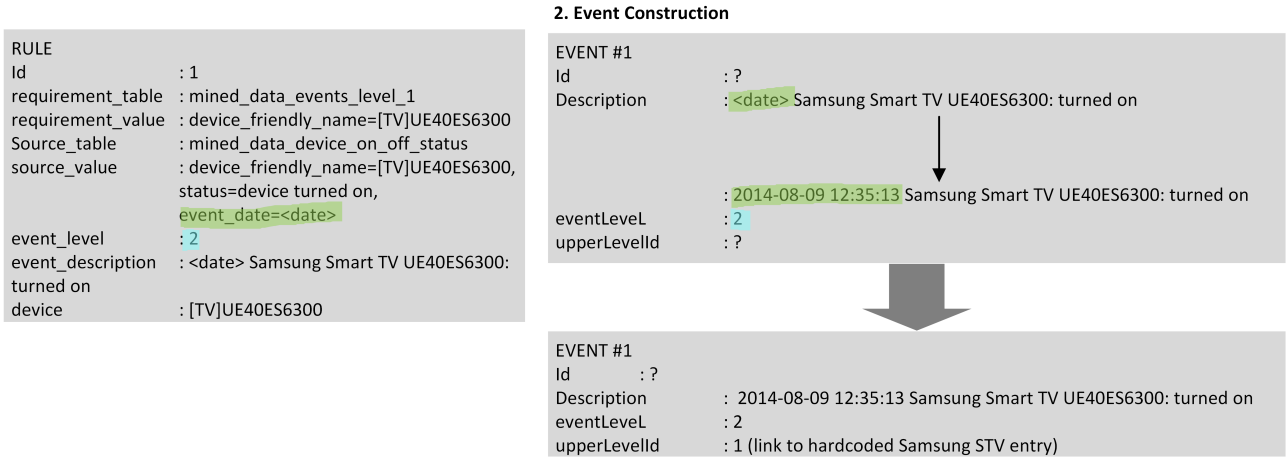



**Figure 38:** Rule Engine Example - Part 2: Event Construction

A requirement check is executed in the third step of the EPU procedure (see Figure 39). This requirement check is performed for each of the previous produced events. The check consists of finding an appropriate event of the next higher event level. A list of all fitting upper level events can be requested from the local database using the *requirement_table* and the *requirement_value* of the current rule. If more than one appropriate upper level event is existing, the event with the closest date and time is chosen. If no upper level event exists, the constructed event is dropped. After finding a fitting upper level event, the related event ID is filled in the field *Upper Level ID*. In the end, the constructed event is saved in the local database.

In the example, an upper level event is searched for each event of step two. For each of the 34 events created before, an upper level event is searched. A SQL statement is constructed under the use of *requirement_table* and *requirement_value* as similar as the statement of the source information. The query from Figure 39 is *SELECT * FROM mined_data_events_level_1 WHERE device_friendly_name=[TV]UE40ES6300*. Due to the fact that the level 1 events are hardcoded, all 34 events are assigned the event ID from the STV test device entry. Afterwards, all events are forwarded to the local database and stored in table *mined_events_level_2*.

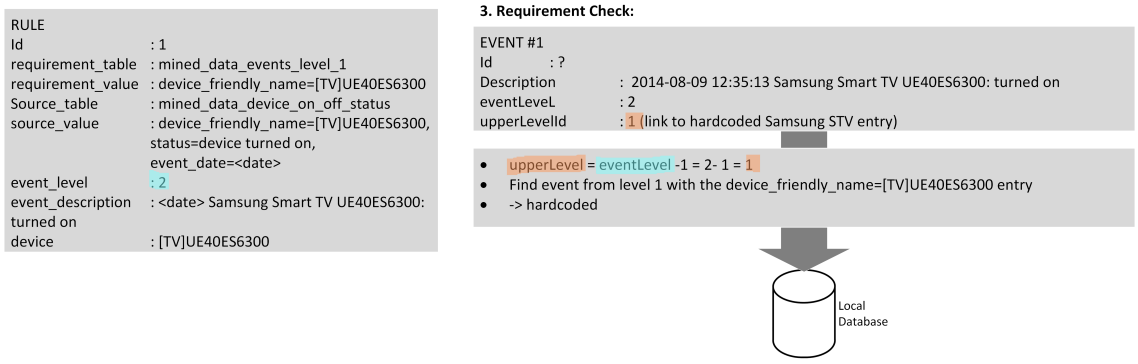



**Figure 39:** Rule Engine Example - Part 3: Requirement Check

### 3. Process Events

The third functionality implemented in the Data Mining Agent will be described in the following paragraph. This part of the Data Mining Agent is responsible for cleaning the produced events. The data cleaning runs through all events level 3. It detects doubled events and reduces so the amount of appearing level 3 events. In the following, it is described how the cleaning works.

**Doubled events** are events that occur more than once. They have the same *Description*, *Event Level* and *Upper Level Id*. Doubled events occurring often in context of measuring network traffic. For example, HTTP messages that are sent or received one after another in a short time interval. The Data Mining Agent goes through the whole list of level 3 events and searches for doubled events. If the Data Mining Agent finds doubled events, all events belonging to these are removed. A new event is created with the same parameters of the doubled events but with an additional index in the event description. This index shows the amount of doubled entries. The Network Analyzing Agent receives for example three packets with the name „HTTP Request“. The Data Mining Agent compromises these three packets to one event with the name „3x HTTP Request“.

### 4. Mine Additional TV Program Information

Within this part of the Data Mining Agent additional TV program information is requested from a website. The additional program information is used to receive more details on the running program. Therefore, a specific website is requested[16].

The additional TV program information consists of the six different aspects. Each aspect is listed in Table 14 and illustrated with an appropriate example.

| Name Of Information Part | Example |
|---|---|
| Name of Program | Tagesschau |
| Major Category | Information |
| Minor Category | News |
| Program Description | Tagesschau is the best address to obtain news and information each day. (...) |
| Program Start | 2014-08-20 20:00:00 |
| Program End | 2014-08-20 20:15:00 |

**Table 14:** Additional Program Information - Example

*Name of Program* is the title of the running program. *Major Category* is the name of the major category the program is belonging to. The major category can be one of six different names: information, series, sports, movie, show and kids. *Minor Category* is the Name of the minor category the program is belonging to. The minor category is a sub category of the major category describing the TV program in more detail. *Program Description* is a descriptive text with a short comprehension of the TV program. *Program Start and End* consists of the beginning and ending date of the running program.

The additional program information is requested for each level three event containing the key word *channel* in the event description. All requested information is forwarded to the local database. The database entry consists of the event ID in order to reference it to the source event.

[16] `http://www.hoerzu.de/text/tv-programm/index.php`

One problem occurres while using the web page mentioned above. The program information is only available for two days. This means, the additional program information has to be requested at least every second day ensuring that all information available is requested.

### 4.2.5 User Interface

The implemented user interface of the PriSEMD prototype is described in this section. Due to time issues, it was not possible to implement the complete functionalities described in the PriSEMD context 3.1.4. The implementation itself is limited to the several visualizations. There are two different data visualizations made, the protocol view and the diagram view. Both are introduced in the following subsections.

#### Protocol View

The protocol view shows the complete event hierarchy of the PriSEMD prototype in a structured list. This list is divided into three different stages representing the three levels of the event structure. First, the level 1 events are displayed. While clicking on the triangle next to a specific event, the next higher event level stage occurs. Clicking on the triangle of a level 1 event makes for example all events level 2 visible that are belonging to the specific level 1 event.

Figure 40 is a screenshot of PriSEMDs protocol view showing events from level one. The screenshot shows two entries, *Device identified: [TV]UE40ES6300* and *Device identified: XBox 360*. The entry belonging to the STV test device consists of the data collected within the example test. The XBox entry is added in order to visualize that other devices can be easily extended to the protocol view. The XBox entry has no further events.

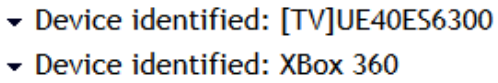



**Figure 40:** Protocol View: Events From Level 1

While clicking on the triangle next to the *Device identified: [TV]UE40ES6300* entry, all level 2 events belonging to the STV event are occurring. Figure 41 shows the protocol view after clicking on the triangle. All events from level 2 are visible now.

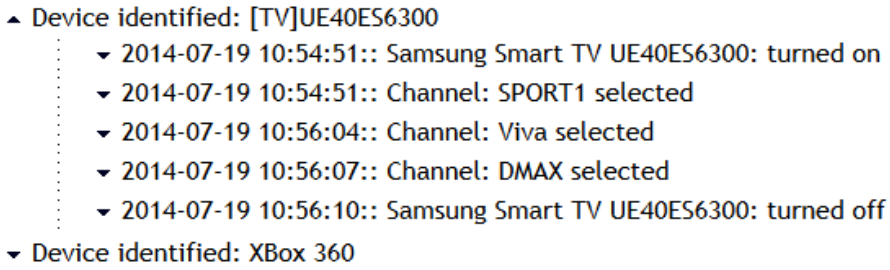



**Figure 41:** Protocol View: Events From Level 2

Figure 42 shows the level 3 events from the first two events level 2.

#### Diagram View

The diagram view consists of three different diagrams. Each diagram summarizes data that is produced in the Analysis Phase of PriSEMD. Besides the diagrams a log entry exists for each diagram. The log entry consists of the data base used to create the diagram. All effort that is needed to create the diagrams and log entries is done by a python script. This script requests the all required data from the local database and calculates the desired log file and diagram images. For the diagrams a special library called *Pygal*[17] is used.

The following figures are produced from an example analysis of the PriSEMD prototype. Therefore, the prototype was executed for one day in a test environment containing a STV test device.

The first Diagram shown in Figure 43 consists of a statistic which channel was watched most. A percent value is assigned to each channel. As shown in Figure 43 the channel *DMAX* is watched most with a percent value of *26.35%*. The data base in the gray box was used for creating the diagram. The *occurrence counter* values are calculated while counting the occurrence of each channel in the Data Mining Agent's *additional program information* data ( 4.2.4 - 4. Mine Additional Program Information). The *occurrence counter* values are absolute values, automatically transformed to appropriate percentage values by Pygal.

[17] http://pygal.org/

- Device identified: [TV]UE40ES6300
  - 2014-07-19 10:54:51:: Samsung Smart TV UE40ES6300: turned on
    - 2014-07-19 10:54:20:: Key: KEY_POWER pressed
    - 2014-07-19 10:54:51:: 6x HTTP Request
  - 2014-07-19 10:54:51:: Channel: SPORT1 selected
    - 2014-07-19 10:54:51:: 2x HTTP Response
    - 2014-07-19 10:54:51:: HbbTV Request
    - 2014-07-19 10:54:51:: Volume: 10
    - 2014-07-19 10:54:51:: Mute:: off
    - 2014-07-19 10:56:03:: Key: KEY_CHANNEL_UP pressed
  - 2014-07-19 10:56:04:: Channel: Viva selected
  - 2014-07-19 10:56:07:: Channel: DMAX selected
  - 2014-07-19 10:56:10:: Samsung Smart TV UE40ES6300: turned off
- Device identified: XBox 360

**Figure 42:** Protocol View: Events From Level 3

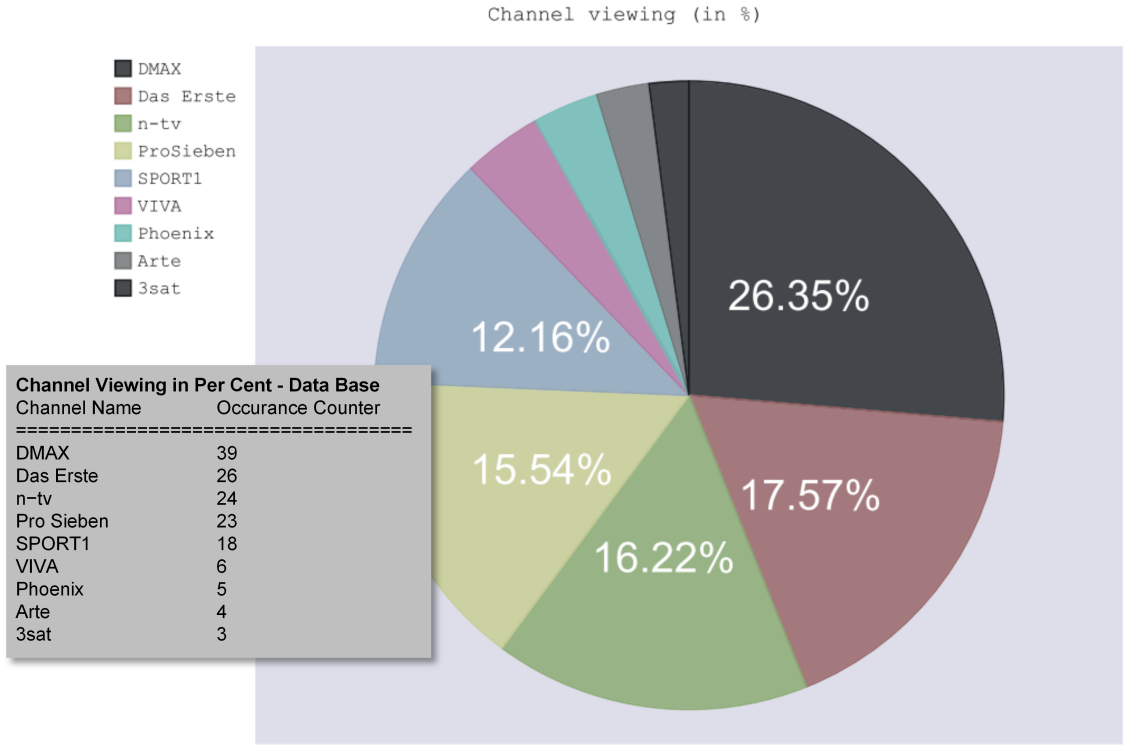



**Figure 43:** Channel Viewing in Per Cent

Figure 44 shows the TV running time per day. Due to the fact that only one day was captured within the example test, values for two further days are inserted in order to obtain a good visualization. As shown in data base box there are three intervals of TV viewing. The first bar corresponds to the first entry in box, one hour and 53 minutes are 113 minutes in the diagram. The data base used for this diagram is created from the Data Mining Agent's *extract device status* data ( 4.2.4 - 1. Extract Device Status). The intervals where the SED was running are summed up for each day.

Figure 45 illustrates information about the watched TV program. Therefore, the major and minor categories of the watched TV program are displayed. On the edge of the pie diagram the minor categories and in the middle the major categories are displayed. As described in Chapter 4.2.4 there are six different major categories. One major category is missing in Figure 45, because it was not watched in the captured time interval. Each major category is divided into several minor categories. The major category *Serie* can be for example divided into four minor categories *Dokumentation, Geschichtsdoku, Porträt* and *Tierdoku*. The value printed on each slice represents a percent value of how often a program

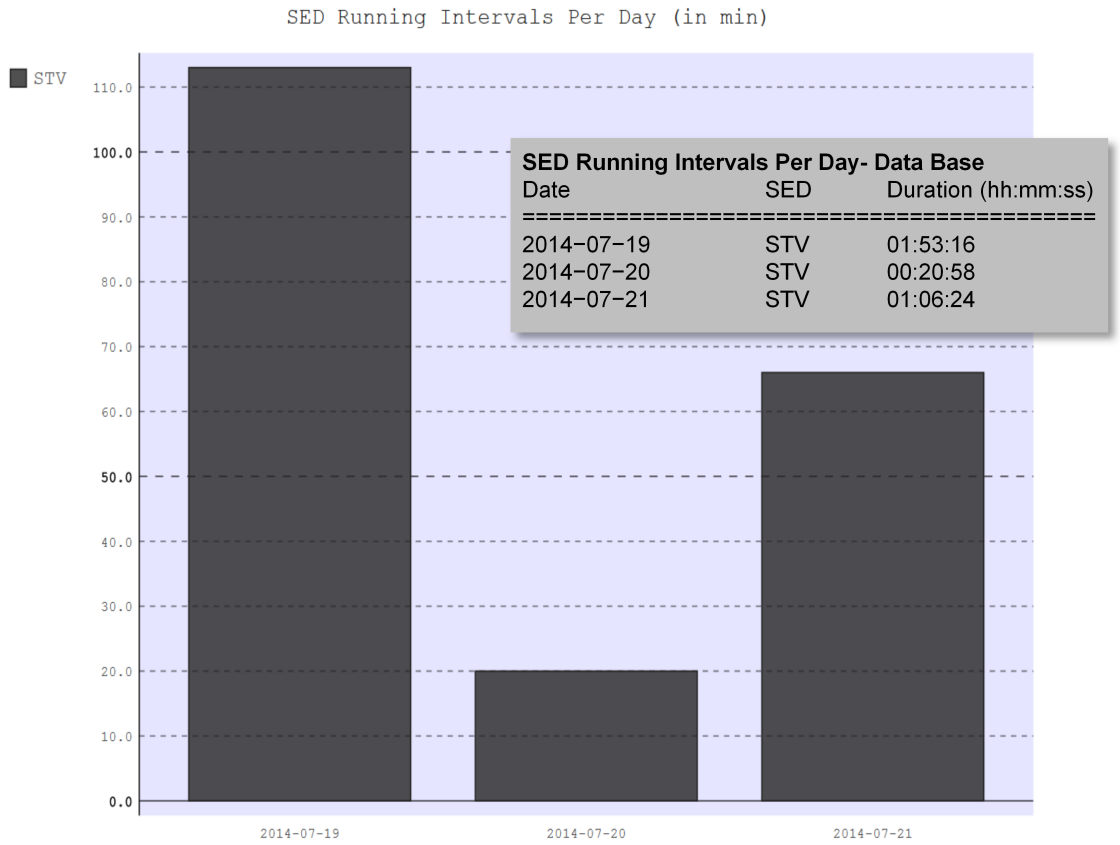
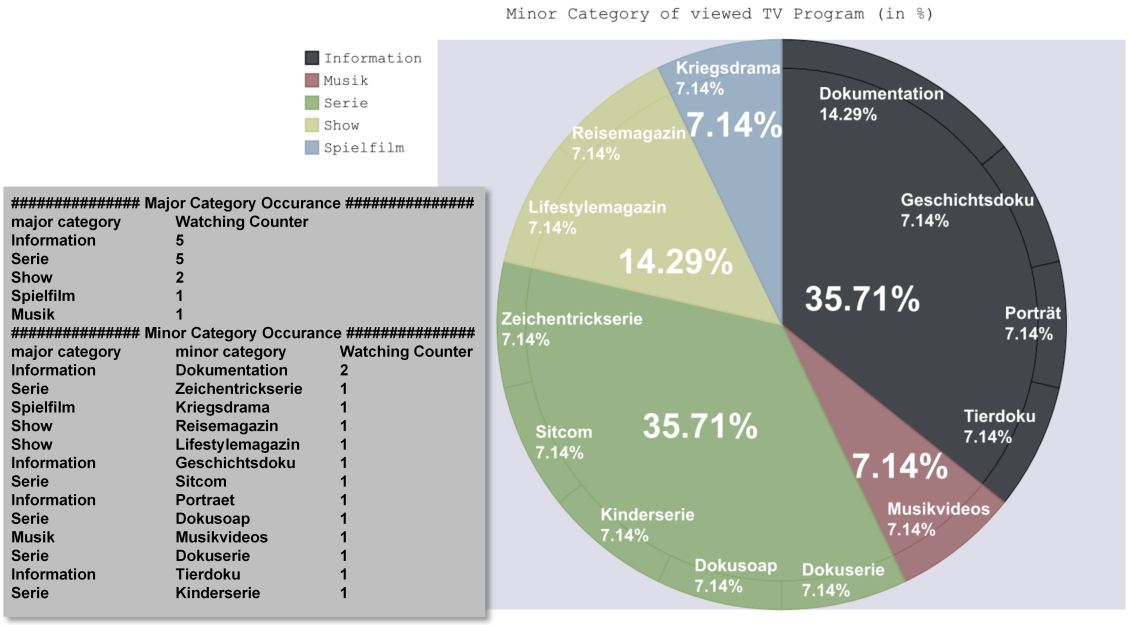



**Figure 44:** TV Running Intervals Per Day

of a specific category was watched. The data base used for the visualization is based on the Data Mining Agent's *additional program information* data ( 4.2.4 - 4. Mine Additional Program Information). The *watching counter* values are calculated while counting the occurrence of each minor or major category. The transformation in percentage values is automatically done by Pygal.




| ############### Major Category Occurance ############### | |
|---|---|
| major category | Watching Counter |
| Information | 5 |
| Serie | 5 |
| Show | 2 |
| Spielfilm | 1 |
| Musik | 1 |

| ############### Minor Category Occurance ############### | | |
|---|---|---|
| major category | minor category | Watching Counter |
| Information | Dokumentation | 2 |
| Serie | Zeichentrickserie | 1 |
| Spielfilm | Kriegsdrama | 1 |
| Show | Reisemagazin | 1 |
| Show | Lifestylemagazin | 1 |
| Information | Geschichtsdoku | 1 |
| Serie | Sitcom | 1 |
| Information | Portraet | 1 |
| Serie | Dokusoap | 1 |
| Musik | Musikvideos | 1 |
| Serie | Dokuserie | 1 |
| Information | Tierdoku | 1 |
| Serie | Kinderserie | 1 |

**Figure 45:** Minor Category Occurrence

Another example for presenting interesting information is shown in Listing 1. In this listing, intervals are shown when the remote control of the STV is being used. One interval consists of several pressed key events. An interval is closed after one minute of detecting no more pressed buttons on the remote control. Using the information in Listing 1 the user activity can be determined. At the given intervals, the user was in front of the STV and interacting with it using the remote control.

```
######################## Infrared Remote Control Usage ########################
# between 2014-07-19 08:59:33 and 2014-07-19 09:00:57 remote keys pressed:18
# between 2014-07-19 10:54:24 and 2014-07-19 10:54:54 remote keys pressed:15
# between 2014-07-19 10:56:03 and 2014-07-19 10:56:12 remote keys pressed:6
# between 2014-07-19 10:58:42 and 2014-07-19 10:59:11 remote keys pressed:4
# between 2014-07-19 11:02:42 and 2014-07-19 11:02:57 remote keys pressed:10
# between 2014-07-19 11:16:54 and 2014-07-19 11:17:09 remote keys pressed:9
# between 2014-07-19 11:33:06 and 2014-07-19 11:33:09 remote keys pressed:5
# between 2014-07-19 11:35:32 and 2014-07-19 11:35:39 remote keys pressed:2
# between 2014-07-19 11:38:56 and 2014-07-19 11:38:56 remote keys pressed:1
# between 2014-07-19 16:41:09 and 2014-07-19 16:42:04 remote keys pressed:6
# between 2014-07-19 17:41:29 and 2014-07-19 17:41:30 remote keys pressed:2
# average key pressure per interval: 7.09090909091
###############################################################################
```

**Listing 1:** Infrared Remote Control Usage

# 5 Evaluation

For evaluation purpose the PriSEMD prototype was tested for a period of one week. A summary of the test results are stated in Section 5.1. Aspects of measuring SEDs in general are mentioned in Section 5.2. Four more SEDs are introduced and a short overview concerning measurable functionalities is provided.

## 5.1 PriSEMD Prototype Evaluation

The PriSEMD prototype evaluation is a data analysis of a one-week-test. The test was started on *Sunday 2014-09-07* and ran until *Sunday 2014-09-14*.

For testing purpose the PriSEMD prototype was embedded into a real household. The network infrastructure of the household consists of the same components and structure as the network environment shown in Figure 30. The environment includes a Samsung UE40ES6300 STV, a Cubietruck hosting the PriSEMD software and a router. During the test interval, the STV was used by a real person in order to watch TV. Further connected to the STV but not measured by PriSEMD are a video gaming console and a Blu-ray player.

The Infrared, Network Analyzer and UPnP Agent were started on Sunday *Sunday 2014-09-07* and ran the entire week. The agents were recording data in the background and saving them in the local database. The **Data Mining Agent** was triggered once a day at 4:30 am. The triggering process was handled by a cron job.

### 5.1.1 Test Results

The data evaluation considers the data from *Monday 2014-09-08* to *Saturday 2014-09-13*. The two other days at the beginning and end of the test interval are left out, because the testing procedure was not running the entire 24 hours.

The database consists of 9233 raw data and 13786 event entries after the test interval. Four different examples of results are shown and explained in the following.

#### Watched TV per Day

The first example illustrates how long TV was watched every day. Therefore, the data produced within the *Mine Device On/Off Status* step of the Data Mining Agent is considered. The time difference between each *turned on* and *turned off* event is calculated and summed up for each day.

The results are displayed in Figure 46. The diagram shows how long the TV was used every day. The longest period of TV watching was reached on Friday the 12-09 with 5 hours 20 minutes. No TV was watched on Wednesday 10-09.

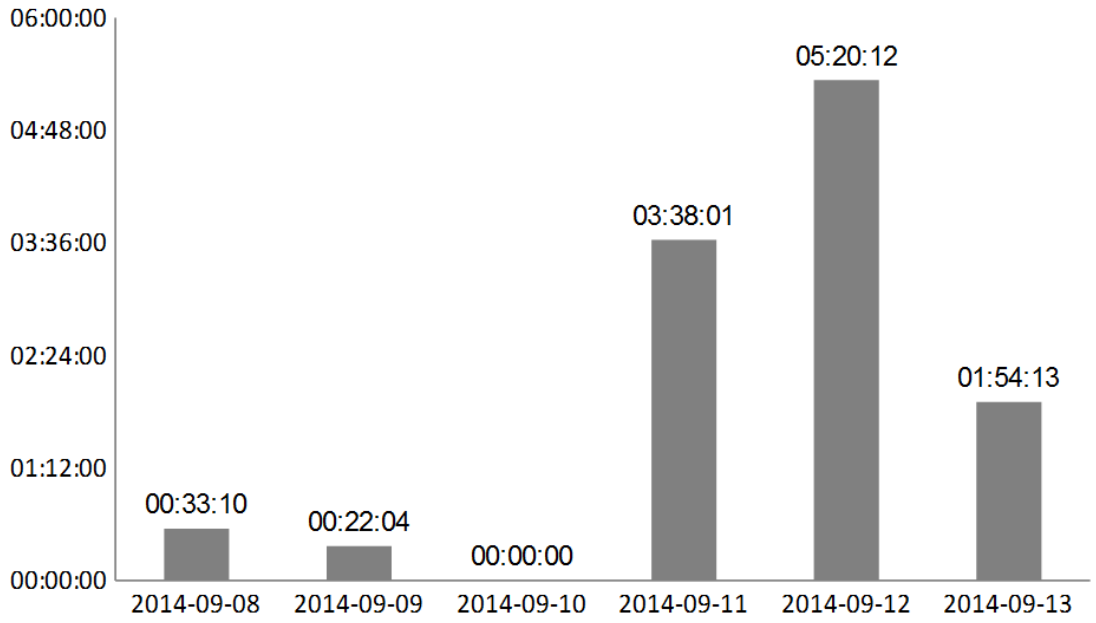



**Figure 46:** Watched TV Per Day

**Favorite Channels**

The second example shows the five most viewed channels during the testing procedure. Here, the Data Mining Agent's results of the *Mine Additional TV Program information* step are considered. The occurrence of each channel in the list of requested program information is counted.

The results are shown in Figure 47. Pro7, RTL, NTV and RTL II are the four most popular channels. All other channels are listed in the group miscellaneous.

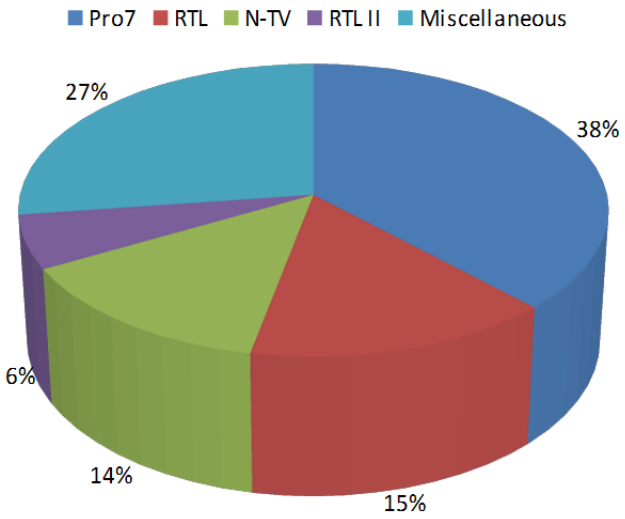



**Figure 47:** Favourite Watched Channels

**Favorite Program Classified Using Categories**

The third example shows which sort of TV program was watched during the seven days of testing. The TV program is categorized using the two category levels, **major** and **minor categories** introduced in the *Mine Additional TV Program information* part of the Data Mining Agent.

Table 15 shows the result of this example. The three most watched major categories are Information, Series and Show. The column „minor categories" shows the most viewed minor categories within the specific major category. Dokureihe was for example watched most within the major category information. All percentage values shown in Table 15 are total values. They describe which part of the total viewed program the specific category consists of. Information is e.g. the most viewed major category with 44%, the minor category Dokureihe has 8%.

| Major Category Name | Percentage | Minor Categories |
|---|---|---|
| Information | 44% | Dokureihe 8%, Dokumentation 6%, Nachrichten 4% |
| Series | 25% | Dokusoap 9% |
| Show | 13% | Show 3%, Werbesendung 2% |

**Table 15:** Favourite Watched Program Categories

**Favorite TV Shows**

The last example illustrates which TV shows are preferred by the test user. For this example the data of the Data Mining Agent's *Mine Additional TV Program information* step are considered. Each entry of Data Mining Agent's result list is counted if the name of the TV show is similar.

The result of this example is shown in Table 16 which consists of the top five watched TV shows. Connecting the results of Table 16 to the example of the most viewed channels, the two most viewed channels Pro7 and RTL are as well represented in the list of top five TV shows.

### 5.1.2 Analysis of Agents

**Raw Data Agents**

The raw data agents are running 24/7 during the complete test interval of seven days. The **Network Analyzer** and **Infrared Agent** are saving a result entry each time either an appropriate network packet or an infrared signal occurs.

| Ranking | Name of Program | Channel |
|---|---|---|
| 1 | How I Met Your Mother | Pro 7 |
| 2 | The Big Bang Theory | Pro 7 |
| 3 | Die Trovatos - Detektive decken auf | RTL |
| 4 | Two and a Half Men | Pro 7 |
| 5 | Nachrichten | multiple channels |

**Table 16:** Favorite TV Program

The **UPnP Agent** requests the UPnP services of the STV in a time interval of five seconds. If the STV is active, the results are saved in the local database. Otherwise, the request throws an exception and becomes cancelled. No data is saved in the database concerning this case.

Concerning runtime, no problems are occurred during the test. Each raw data agent ran from the start to the end without crashing.

The CPU load produced by the raw data agents is examined hereafter. Thus two situations are distinguished. A high workload situation is for example when each raw data agent is receiving input at the same time, a low workload situation where none of the agents receive anything. The CPU load is slightly higher in high workload situations than in situation where the workload is low. Concerning the raw data agents, the CPU load is generally low.

#### Data Mining Agent

As stated in the introduction of this chapter, the **Data Mining Agent** is triggered once a day at 04:30 am. Within the complete test, the Data Mining Agent was triggered seven times. Each time, the Data Mining Agent was executed, raw data from the day before is analyzed. Executing the Data Mining Agent at 2014-09-10 04:30:00 leads to the analysis of the raw data produced from 2014-09-09 04:30:00 to 2014-09-10 04:29:00.

There are several reasons why the Data Mining Agent was triggered as described above:

- One possibility to keep the calculation time low is to start the Data Mining Agent periodically. The execution time of the Data Mining Agent depends on the available raw data that has to be analyzed. Hence, the Data Mining Agent was triggered once a day. The raw data produced by the STV test device can be analyzed in under 30 minutes. The execution times of the Data Mining Agent are displayed in Table 18 and will be described later in this section.

- Another reason why the Data Mining Agent was executed once a day is the availability of the TV program information. The requested data within the *Mine Additional TV Program information* step are only available for two days in the past.

- The Data Mining Agent is started in the early morning hours, because this is the time where the test user usually watches no TV. Therefore, the Data Mining Agent can analyze the raw data without new input of the raw data agents. During the procession of the Data Mining Agent, the raw data agents shall be paused or shall not measure new input. It is possible that new input measured in the execution of the Data Mining Agent is not considered in the event calculation process.

The Data Mining Agent produces heavy CPU load in contrast to the raw data agents.

#### Analysis of Data Produced by the Agents

Table 17 shows how much event and raw data entries are inserted in the database day by day. On *2014-09-10* the Network Analyzer Agent inserts 215 entries in the local database. However, no TV was watched on that day, shown in Figure 46. The Infrared and UPnP Agent as well as the Data Mining Agent were not inserting entries in the database. Due to the fact that network traffic was measured, it can be expected that the STV was using the network interface in standby mode, for example for installing a firmware update. The Data Mining Agent builds no events for that day. Level two events cannot be built, because appropriate raw data is missing. The recorded network data can only being used to build level 3 events. Due to the fact that level 2 events are missing, the level 3 events cannot be built, because the connection to the upper level event is missing.

The mine additional device functionality of the Data Mining Agent as well as the UPnP Agent produced no entries on *2014-09-12* and *2014-09-13*.

Table 18 shows the execution time of the Data Mining Agent concerning each day. The date describes the day where the analyzed data was measured, the Data Mining Agent was triggered on the next day. For example the measured interval from *09-08 04:30:00* to *09-09 04:29:00* is labeled with the date *09-08*. On that day, the STV run 33 minutes and

| Event Entries | 09-08 | 09-09 | 09-10 | 09-11 | 09-12 | 09-13 | total |
|---|---|---|---|---|---|---|---|
| level 2 events | 385 | 132 | 0 | 412 | 74 | 36 | 1039 |
| level 3 events | 3900 | 551 | 0 | 3347 | 1512 | 508 | 9818 |
| device status | 105 | 66 | 0 | 232 | 91 | 170 | 664 |
| additional program information | 997 | 422 | 0 | 846 | 0 | 0 | 2265 |
| Data Mining Agent | 5387 | 1191 | 0 | 4928 | 1677 | 714 | 13786 |
| Raw Data Entries | | | | | | | |
| Infrared Agent | 50 | 27 | 0 | 151 | 104 | 114 | 446 |
| Network Analyzer Agent | 1591 | 638 | 215 | 2345 | 2236 | 1632 | 8657 |
| UPnP Agent | 41 | 10 | 0 | 79 | 0 | 0 | 130 |

**Table 17:** Entries Inserted in Database Day by Day

10 seconds. The Data Mining Agent triggered at *09-09 04:30:00* needs 5 minutes and 18 seconds to analyze the data produced within the previous stated interval.

Table 18 shows the execution time of the Data Mining Agent for each day of the test. In addition, the watched TV time is added in the second row. A comparison of Table 18 and Table 17 shows that the execution time of the Data Mining Agent is depending on the amount of raw data entries. The Data Mining Agent was executed 2 minutes and 54 seconds on *09-10* where the TV was not turned on by the user. A very low amount of raw data has to be analyzed and no events are produced. Furthermore, no additional program information was requested. The approximately 3 minutes can be determined as minimal runtime for the Data Mining Agent.

| Watched TV | 09-08 | 09-09 | 09-10 | 09-11 | 09-12 | 09-13 |
|---|---|---|---|---|---|---|
| DMA Runtime | 5:18 min | 4:34 min | 2:54 min | 9:19 min | 8:23 min | 11:20 min |
| Watched TV | 00:33:10 | 00:22:04 | 00:00:00 | 03:38:01 | 05:20:12 | 01:54:13 |

**Table 18:** Watched TV and DMA Execution Time Per Day

### 5.1.3 Database Space Analysis

A prediction of the estimated database size and a discussion of data compressing strategies are provided in this section.

The size of a raw data entry is assumed to be 200 Byte, the size of an event entry to be 270 Byte. During the seven day test 9233 raw data entries and 13786 event entries are stored in the local data base. The size of the raw data entries per day can be calculated as follows:

$$\text{size(raw-data) per week: } 200Byte * 9233 = 1846,6kByte$$

$$\text{size(raw-data) per day: } 200Byte * 9233/7 = 263,8kByte$$

The size of the event entries per day is calculated hereafter:

$$\text{size(events) per week: } 270Byte * 13786 = 3722,2kByte$$

$$\text{size(events) per day: } 270Byte * 13786/7 = 531,746kByte$$

Table 19 shows the database size for one month and year are linearly extrapolated. After one year measuring the STV test device, the local database size can be approximately determined to 290,374 MByte. Compressing data is not required while measuring one SED. However, including more than one SED in the measurement process leads to the fact that the size of the database will grow faster. The compression could be established in deleting all raw data entries after one month, because they are no more needed for event calculation purpose. Compressing the event entry data could be established in creating an overview about all important information and deleting the events entries. Events of received network packets can be e.g. summarized while counting the amount of received packets. For example 150 HTTP request event entries could be replaced by one entry in the overview stating that 150 HTTP packets are received.

A problem going along with the data compression is that the overview has fewer details than the event entry representation. In the end, it has to be considered in each case which representation is more advantageous.

## 5.2 Measuring Interfaces in Practice

The measurement of the network, UPnP and infrared interfaces for general SEDs are examined in this section. After investigating the STV test device, this chapter focuses on measuring data in general, not of a specific SED.

| | 1 Day | 1 Week | 1 Month | 1 Year |
|---|---|---|---|---|
| raw data | 263,8 KByte | 1,847 MByte | 7,914 MByte | 96,287 MByte |
| events data | 531,746 KByte | 3,722 MByte | 15,952 MByte | 194,087 MByte |
| total DB size | 795,546 KByte | 5,569 MByte | 23,866 MByte | 290,374 MByte |

**Table 19:** Database Summary

### 5.2.1 Network Interface Measurement in Practice

The analysis of network traffic enables measuring general facts of any SEDs. Information can be obtained while analyzing the network packets sent or received by SEDs. The following listing shows which indicator can be used for which kind of information retrieval:

- **Network activity is an indicator for SED activity**: Network activity means actively sending and receiving network packets. A SED is running if it sends and receives network packets, otherwise it is turned off. As stated in the previous Section 5.1.2, the test device is communicating via network while it is in standby mode, e.g. for installing and downloading an update. Such situations can be determined while checking if other agents are measuring input. If all other agents receiving no input, it is very likely that the SED is in standby mode.
- **SSDP and DHCP packets can be used to determine the SED status**: Receiving network packets using either the SSDP or DHCP protocol can be used to determine when a device was either in the *on*, *running* or *off* status. The analysis of SSDP and DHCP was mentioned in Chapter 3.2.1.
- **Analyzing the IP section can be used to determine communication partners**: Each network packet using the IP protocol delivers two IP addresses. One IP address belongs to the SED, the other to its communication partner. A tool like *nslookup* can be used to resolve the IP address to a human readable name.
- **The SED can be identified using IP, MAC and SSDP**: The combination of IP and MAC address can be used to create a unique identifier allowing PriSEMD to differentiate SEDs. SED device, model and manufacturer name can be obtained analyzing SSDP packets.
- **Analyzing HbbTV packets delivers channel information**: The analysis of HbbTV packets can be used to track STV channel information. Therefore, the URL containing in the HbbTV packet has to be parsed in order to obtain the channel name. The timestamp when the packet was received and the channel name can be used to determine when a specific channel was watched. This attempt assumes that the SED supports HbbTV and is usually a STV or a set-top-box.

### 5.2.2 UPnP Interface Measurement in Practice

Requesting UPnP services allows the measurement of various information. For all practical purposes, the amount of available information depends on the manufacturers service implementation. The UPnP implementation can vary enormous from manufacturer to manufacturer and even between different model series of the same manufacturer. The two STVs displayed in Table 20 are an example for a very different UPnP implementation. As mentioned in Chapter 4.1.1 the Samsung UE40ES6300 STV has a lot of different UPnP services. In contrast to the test device, the Sony Bravia STV has only a few implemented UPnP functionalities. Besides getting and setting the current volume and turning on and off the mute status there are no other useful services implemented. The UPnP interface is a non-reliable information source, because the differences between manufacturers' implementations are too large. Nevertheless, the UPnP interface can be used as an additional source of information. It remains to be seen how the future develops the UPnP service availabilities.

### 5.2.3 Infrared Interface Measurement in Practice

This interface allows the measurement of infrared signals produced by remote controls. The following two aspects can be determined while analyzing these signals:

- **Receiving an infrared signal is an indicator for user activity**: Each time an infrared signal is received, the user has to trigger the specific button on the remote control. It may be assumed that the user stands in front of the SED and interacts with the device.
- **The signal code can be mapped to the name of the pressed button**: Each button on the remote control has its own signal code. Using the LIRC tools explained in Chapter 4.2.1 the signal code can be mapped to a previous defined button name.

In practice, there is a problem to deal with. Different remote control models are having different buttons and sending different signal codes. A standardized button to signal mapping does not exist. This problem can complicate finding out which button was pressed on the remote control. However, this problem can be neglected for detecting the user activity. It is important to receive any signal, it does not matter what specific signal is received.

### 5.2.4 Classification of SEDs in Practicse

The STV test device is compared with four other SEDs in this section, the video gaming consoles XBox 360 and XBox One, the STV Samsung UE37ES5700 and the STV Sony Bravia KDL-40HX855. Table 20 summarizes the available interfaces concerning PriSEMD data measurement. Furthermore, the test device Samsung UE40ES6300 is added in order to complete the summary.

| SED Category | Name | Characteristics |
|---|---|---|
| video gaming console | Microsoft XBox One | network, SSDP |
| video gaming console | Microsoft XBox 360 | network, SSDP, UPnP |
| STV | Samsung UE40ES6300 | network, SSDP, UPnP, Infrared, HbbTV |
| STV | Samsung UE37ES5700 | network, SSDP, UPnP, Infrared, HbbTV |
| STV | Sony Bravia KDL-40HX855 | network, SSDP, UPnP, Infrared, HbbTV |

**Table 20:** SED Summary

Each SED is introduced with a short description in the listing below. The description outlines measurable interfaces provided by the SED and a comparison to the test device focusing the main differences concerning PriSEMD data measurement.

- **Microsoft XBox 360**: Microsoft's video gaming console is sold since 2005. The XBox 360 provides a network interface which is used for example to surf in the Internet or to receive updates from the manufacturer. The XBox 360 supports UPnP and SSDP. The available UPnP services are restricted for example to get/set volume or toggle mute status. The implementation of the SSDP protocol differs in one point from the test device. The SSDP alive messages are only sent once at boot time of the XBox 360. A renewal of the alive status does not take place. An infrared remote control can be purchased additionally. That's why the infrared interface is not considered in the summary of Table 20 [51].

- **Microsoft XBox One**: The XBox One is the successor product to the XBox 360 and is sold since 2013 [51]. The video gaming console provides a network interface and supports SSDP. In contrast to the XBox 360 no UPnP server is implemented. The SSDP implementation is very different to the implementation of the XBox 360 and the test device. The XBox One performs a m-Discovery search during the boot cycle and in periodically intervals while it is running. SSDP alive and SSDP byebye messages are not sent from the XBox One. An infrared remote control can be purchased additionally for the XBox One as well.

- **Samsung UE37ES5700**: The Samsung UE37ES5700 is a STV from the similar ES model series as the test device. The available interfaces for the Samsung UE37ES5700 are the same as for the test device UE40ES6300.

- **Sony Bravia KDL-40HX855**: The Sony Bravia KDL-40HX855 is a STV having network, UPnP, SSDP, HbbTV and Infrared. In contrast to the test device, the UPnP interface of the Sony Bravia STV supports fewer services. Besides get/set volume and toggling the mute status it is for example possible to send infrared codes in order to simulate a remote control button pressure. The SSDP implementation is similar to the test device's implementation. Sending SSDP-alive packets at boot time and periodically during the device is running and sending SSDP-byebye packets when the STV is turned off.

# 6 Conclusion and Outlook

## 6.1 Future Work

This subsection includes topics for extending the work of this thesis. The PriSEMD concept is a general concept for measuring, analyzing and presenting data of SEDs. The PriSEMD prototype is a software which has implemented several basic functionalities of the PriSEMD concept. The hereafter presented issues can be used to extend both, the PriSEMD concept and the implemented prototype.

The local user acceptance of data transmission could be increased by offering a payment model. Jon Kleinberg et al. are stating in their paper [31] that users who benefit from data should compensate users who provide data. Concerning PriSEMD, the local user has no right motivation allowing data transmission to third party users. A payment model where the local user is paid for shared content could offer an incentive. An assumed payment model could look like follows: The user is paid with a small amount of cash for sharing a specific category of data for a defined time interval. The user obtains e.g. 0,50€ for approving data concerning SED activity for an interval of one month. An overview page included in the PriSEMD GUI could show the profitability of each data category. The local user can inspect the payment offerings and has the possibility to decide if he would like to share data with third parties.

The SED identifying procedure where a unique ID is assigned to each SED could be improved using machine learning. As explained in Chapter 3.2.1, the unique ID is built using MAC and IP address of the device. Both addresses could be changed easily, e.g. via manipulating. Algorithms of machine learning could be used to create an ID while analyzing the SED's behavior. One possibility is to classify SEDs according to their produced network traffic. After a predefined training interval, each SED can be identified without using the addresses stated above.

Another interesting extension could be to detect different users of a SED. The results of analyzing using behavior can be used to determine which user interacts with the SED. Analyzing the SED activity status or content running on the SED can be used for example to classify which user is using the SED currently. Detecting multiple users could be a difficulty going along with this extension. It is more difficult to detect multiple users than detecting single users, because compromises are made concerning usage. One TV program has e.g. to be chosen by more than one person.

The PriSEMD prototype could be extended with several issues:

- The set of measured interfaces could be extended with other common used interfaces, e.g. Bluetooth. Bluetooth is used by a lot of SEDs for embedding third party hardware. There are for example headsets, keyboard/mouse combinations, and 3D glasses that can be connected to compatible SEDs. An extension of the measured interface set allows capturing more details about the user behavior, because more input devices are considered.

- Due to time issues, the functionalities of the PriSEMD concept are not completely implemented within the prototype. One missing aspect concerns the Presentation Phase 3.1.4. The user interface could be extended with access regulation for local and third party user and the approval system which tags the data produced by PriSEMD. Furthermore, the anonymous data transmission concept presented in Chapter 3.3.3 was not implemented. This concept allows third party users to receive data by respecting the privacy of local users.

- As shown in Chapter 4.2.4, the Data Mining Agent needs time to analyze the measured raw data. Analyzing data measured from one day could take around 3 to 12 minutes. Within the analysis procedure ideally no new raw data has to be measured by PriSEMD. This problem could be solved in splitting up the functionality. Two different devices, one for SED measurement and one for analysis purpose. The interval of analysis could be reduced using this approach, e.g. analyzing the measured data each 30 minutes. This would open the possibility to transmit data to third parties in shorter intervals.

- Another aspect of extending the implemented prototype is supporting more SEDs. The prototype measures only the STV test device. An extension could be to generalize the data measurement and support multiple SEDs - ideally all available SEDs in the network.

The evaluation results of the data collected by PriSEMD could be improved by carrying out a larger study. This study shall consist of multiple households measured in a time span of e.g. one month. The results obtained within this study are more informative than the results of the seven day test done in the context of this thesis. The results can be e.g. used to determine more details concerning which aspects of the user behavior are analyzable out of the raw data.

## 6.2 Conclusion

A short summary of the results is presented in the final conclusion. Afterwards, the achieved results are compared to the leading questions stated in the beginning of this thesis.

The PriSEMD concept outlined in this thesis is an audience and environment measurement system for SEDs. The local user can inspect the measured data of PriSEMD and has the possibility to control data sharing with other parties, e.g.

advertising companies or SED manufacturers. This presentation of PriSEMD data shall increase the local users awareness which data can be gathered by SEDs.

The sharing aspect of PriSEMD respects the privacy of the local user. For a third party user it is not possible to identify a local user while analyzing shared content. A compromise between local and third party users is pursued. On the one hand, PriSEMD saves the privacy of local users, on the other hand, data can be transmitted to third parties in order to respect their interest in analyzing using behavior for e.g. enhancing products and services.

The PriSEMD concept demonstrates that SED data measurement is technically realizable. For this purpose, PriSEMD emphasizes how measurement can be performed, how measured data can be analyzed and processed, and how results can be presented. As an example for executing the measurement, the three interfaces infrared, network traffic and UPnP services are explained in detail.

The data obtained by the measurement procedure can be categorized in three groups. First, information about the available SEDs can be gathered. Manufacturer name or model number of a SED can be for example extracted while analyzing the network traffic of a SED. Second, the consumed content can be measured. As an example, requested websites or watched TV program can be requested from special UPnP services. Third, information about the using behavior can be obtained. It is possible to determine when a device is running and when a user interacts with it, for example analyzing Infrared signals or network traffic.

The PriSEMD prototype is an implementation of the PriSEMD concept demonstrating that it is possible to realize the strategies of the concept in an applied software. Furthermore, the evaluation of a 7-day-test confirms that PriSEMD measurement is working in a realistic environment. The results are showing that it is possible to create a detailed profile of the user.

PriSEMD is a new audience measurement strategy offering content providers and manufacturers more information about the usage of SED than currently used audience measurement systems. A s difference to other audience measurement systems is the involvement of current technologies, e.g. Internet. In addition, PriSEMD is a holistic approach considering the measurement of all SEDs available within a computer network.

## References


[1] Erdem U. Altinyurt. *SamyGO*. URL: http://www.samygo.tv/ (visited on 10/28/2014).

[2] Christoph Bartelmus. *What is LIRC ?* URL: http://www.lirc.org (visited on 06/23/2014).

[3] Brian Benchoff. *Getting root on a Sony TV*. URL: http://hackaday.com/2012/06/20/getting-root-on-a-sony-tv/ (visited on 09/28/2014).

[4] Tillmann Braun. *Smart Home entwickelt sich zum Milliardenmarkt*. URL: http://www.zdnet.de/88209177/smart-home-entwickelt-sich-zum-milliardenmarkt/ (visited on 11/02/2014).

[5] T. Breyer-Mayländer and A. Werner. *Handbuch der Medienbetriebslehre*. Oldenbourg, 2003. ISBN: 9783486273564. URL: http://books.google.de/books?id=MI-SPVAquGgC.

[6] Chip.de. URL: http://www.chip.de/news/Samsung-WF12F9E6P4W-Die-WLAN-Waschmaschine_63141257.html (visited on 07/25/2014).

[7] Chip.de. URL: http://www.chip.de/news/LG-Homechat-Kuehlschrank-Co-per-Handy-steuern_66306267.html (visited on 07/25/2014).

[8] Diane J. Cook and Sajal K. Das. *Smart environments - technology, protocols and applications*. Wiley, 2005. ISBN: 978-0-471-54448-7.

[9] Cubieboard.org. URL: http://docs.cubieboard.org/products/start (visited on 07/25/2014).

[10] Jennifer Savage Daniel Crowley and David Bryan. URL: https://media.blackhat.com/us-13/US-13-Crowley-Home-Invasion-2-0-WP.pdf (visited on 08/20/2014).

[11] Oxford Dictionaries. URL: http://www.oxforddictionaries.com/definition/english/privacy (visited on 08/20/2014).

[12] R. Droms. *Dynamic Host Configuration Protocol*. RFC 2131. IETF, 1997. URL: http://www.ietf.org/rfc/rfc2131.txt (visited on 08/14/2014).

[13] George Drosatos, Aimilia Tasidou, and Pavlos S. Efraimidis. „Privacy-Preserving Television Audience Measurement Using Smart TVs.“ In: *SEC*. Ed. by Dimitris Gritzalis, Steven Furnell, and Marianthi Theoharidou. Vol. 376. IFIP Advances in Information and Communication Technology. Springer, 2012, pp. 223–234. ISBN: 978-3-642-30435-4. URL: http://dblp.uni-trier.de/db/conf/sec/sec2012.html#DrosatosTE12.

[14] Cynthia Dwork. „Differential privacy“. In: *in ICALP*. Springer, 2006, pp. 1–12.

[15] ETSI. *Hybrid Broadcast Broadband TV - Technical Specification v1.2.1*. Whitepaper. ETSI. URL: http://www.etsi.org/deliver/etsi_ts/102700_102799/102796/01.02.01_60/ts_102796v010201p.pdf.

[16] Claudia Eckert. *IT-Sicherheit - Konzepte, Verfahren, Protokolle (5. Aufl.)* Oldenbourg, 2008, pp. I–XIV, 1–925. ISBN: 978-3-486-58270-3.

[17] Claudia Eckert. *IT Sicherheit - Konzepte Verfahren Protokolle*. 7th ed. Oldenburg Verlag, 2012.

[18] Elektronik-Kompendium. URL: http://www.elektronik-kompendium.de/sites/net/1406201.htm (visited on 07/14/2014).

[19] Shadi Esnaashari, Ian Welch, and Peter Komisarczuk. „Determining Home Users' Vulnerability to Universal Plug and Play (UPnP) Attacks“. In: *Proceedings of the 2013 27th International Conference on Advanced Information Networking and Applications Workshops*. WAINA '13. Washington, DC, USA: IEEE Computer Society, 2013,

pp. 725–729. ISBN: 978-0-7695-4952-1. DOI: 10.1109/WAINA.2013.225. URL: http://dx.doi.org/10.1109/WAINA.2013.225.

[20] R. Fielding et al. *RFC 2616, Hypertext Transfer Protocol – HTTP/1.1*. RFC 4949. IETF, 1999. URL: http://www.ietf.org/rfc/rfc2616.txt (visited on 07/08/2014).

[21] UPnP Forum. *About UPnP Forum*. URL: http://upnp.org/about/what-is-upnp/ (visited on 06/23/2014).

[22] UPnP Forum. *UPnP Device Architecture 1.1*. Whitepaper. UPnP Forum. URL: http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf (visited on 07/08/2014).

[23] Milosz Galazka. *How to change the MAC address of an Ethernet interface*. URL: http://blog.sleeplessbeastie.eu/2013/01/11/how-to-change-the-mac-address-of-an-ethernet-interface/ (visited on 09/28/2014).

[24] Marco Ghiglieri. *I Know What You Watched Last Sunday - A New Survey Of Privacy In HbbTV*. English. Workshop Web 2.0 Security & Privacy 2014 in conjunction with the IEEE Symposium on Security and Privacy. San Jose, California, May 2014.

[25] Marco Ghiglieri, Florian Oswald, and Erik Tews. „HbbTV - I Know What You Are Watching“. In: *13. Deutschen IT-Sicherheitskongresses*. BSI. SecuMedia Verlags-GmbH, May 2013.

[26] Marco Ghiglieri and Erik Tews. „A Privacy Protection System for HbbTV in Smart TVs“. English. In: *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*. Ed. by IEEE. Las Vegas, NV, Jan. 2014, pp. 357 –362. ISBN: 978-1-4799-2356-4.

[27] Geoff Grindrod. URL: http://blog.trendmicro.com/trendlabs-security-intelligence/the-smartification-of-the-home-part-1/ (visited on 08/20/2014).

[28] HbbTV-infos.de. URL: http://www.hbbtv-infos.de/sender/hbbtv-senderliste.php (visited on 08/18/2014).

[29] J.S. Houston. *Cooperative system for measuring electronic media*. US Patent 6,353,929. 2002. URL: http://www.google.com/patents/US6353929.

[30] Y. Y. Goland J. Cohen S. Aggarwal. *General Event Notification Architecture Base*. Draft. IETF, 1998. URL: http://tools.ietf.org/html/draft-cohen-gena-p-base-01 (visited on 08/19/2014).

[31] Jon Kleinberg, Christos H. Papadimitriou, and Prabhakar Raghavan. „On the Value of Private Information“. In: *Proceedings of the 8th Conference on Theoretical Aspects of Rationality and Knowledge*. TARK '01. Siena, Italy: Morgan Kaufmann Publishers Inc., 2001, pp. 249–257. ISBN: 1-55860-791-9. URL: http://dl.acm.org/citation.cfm?id=1028128.1028156.

[32] Gesellschaft für Unterhaltungs-und Kommunikationselektronik GFU. *Smart TVs verändern das TV-Verhalten der Deutschen*. Press Information. URL: http://www.gfu.de/_dbe,news,_auto_9030457.xhtml (visited on 07/11/2014).

[33] Eugen Mikóczy. *Slovak Telekom: "It is really impressive how many HbbTV devices have been deployed in the last two years"*. URL: http://ottworldsummit.com/slovak-telekom-it-is-really-impressive-how-many-hbbtv-devices-have-been-deployed-in-the-last-two-years/ (visited on 10/11/2014).

[34] HD Moore. *Security Flaws in Universal Plug and Play - Rapid 7*. Whitepaper. Security Street - Rapid 7. URL: https://community.rapid7.com/docs/DOC-2150 (visited on 10/13/2014).

[35] Florian Oswald. „SmartTV - Eine Sicherheits- und Datenschutzanalyse von internetfähigen TVs“. MA thesis. TU Darmstadt, 2013.

[36] The GNOME Project. URL: https://wiki.gnome.org/action/show/Projects/GUPnP?action=show&redirect=GUPnP (visited on 08/19/2014).

[37] Franklin Reynolds. *The Ubiquitous Web, UPnP and Smart Homes*. Whitepaper. Nokia Research Center, Cambridge. URL: http://www.w3.org/2006/02/reynolds-paper.pdf (visited on 06/23/2014).

[38] Margaret Rouse. URL: http://whatis.techtarget.com/definition/proxy-server (visited on 08/15/2014).

[39] Axel Kannenberg Fabian Scherschel. *LG Smart-TVs spähen Nutzer aus*. URL: http://www.heise.de/security/meldung/LG-Smart-TVs-spaehen-Nutzer-aus-2051973.html (visited on 10/11/2014).

[40] Fabian Scherschel. *LG verbietet seinen Smart-TVs das Spionieren*. URL: http://www.heise.de/security/meldung/LG-verbietet-seinen-Smart-TVs-das-Spionieren-2065695.html (visited on 10/11/2014).

[41] E.E.D. Sesto et al. *Configurable monitoring of program viewership and usage of interactive applications*. US Patent 6,530,082. 2003. URL: http://www.google.com/patents/US6530082.

[42] Seungjoo Kim SeungJin Lee. *Hacking, Surveilling, And Deceiving Victims On Smart TV*. Presentation. Blackhat Conference. URL: https://media.blackhat.com/us-13/US-13-Lee-Hacking-Surveilling-and-Deceiving-Victims-on-Smart-TV-Slides.pdf (visited on 08/20/2014).

[43] Robert W. Shirey. *Internet Security Glossary, Version 2*. RFC 4949. IETF, 2007. URL: http://tools.ietf.org/html/rfc4949 (visited on 06/20/2014).

[44] Sony. *Sony Sound System Image*. URL: http://store.sony.com/SNYNA_27/pimg/pSNYNA-MHCEC609IP_main_v500.png (visited on 12/10/2014).

[45] Statista. URL: http://www.statista.com/statistics/268857/number-of-smart-tv-households-in-the-us (visited on 07/25/2014).

[46] Statista. URL: http://de.statista.com/statistik/daten/studie/208236/umfrage/prognose-zur-entwicklung-der-smart-tv-haushalte-in-deutschland/ (visited on 07/25/2014).

[47] Andrew S. Tanenbaum. *Computernetzwerke*. 3. überarb. Aufl. München [i.e. Haar] [u.a.]: Prentice Hall PTR, 1997. ISBN: 3-8272-9536-X.

[48] Andrew S. Tanenbaum and David J. Wetherall. *Computer Networks*. 5th. Upper Saddle River, NJ, USA: Prentice Hall Press, 2010. ISBN: 0132126958, 9780132126953.

[49] C. Warner. *Media Selling: Television, Print, Internet, Radio*. Wiley Desktop Editions Series. John Wiley & Sons, 2009. ISBN: 9781405158398. URL: http://books.google.de/books?id=nZqpU0r360UC.

[50] Wikipedia. URL: http://de.wikipedia.org/wiki/Universal_Plug_and_Play#mediaviewer/Datei:Upnp_architecture.svg (visited on 07/02/2014).

[51] Wikipedia. URL: http://en.wikipedia.org/wiki/Xbox_360 (visited on 09/16/2014).

[52] Wikipedia. *Server Name Indication*. URL: http://de.wikipedia.org/wiki/Server_Name_Indication (visited on 11/23/2014).

[53] T.B. Zahariadis. *Home Networking Technologies and Standards*. Artech House telecommunications library. Artech House, 2003. ISBN: 9781580536493. URL: http://books.google.com.au/books?id=die9n5GTnSwC.

[54] Burns et al. *Security Power Tools*. First. O'Reilly, 2007. ISBN: 9780596009632.

[55] ambrosa. *Is LIRC working ?* URL:
https://github.com/cubieplayer/Cubian/issues/75#issuecomment-23358111 (visited on 10/10/2014).

[56] diffen.com. URL: http://www.diffen.com/difference/Data_vs_Information (visited on 07/15/2014).

[57] gawkerassets.com. *Samsung remote control image*. URL:
http://img.gawkerassets.com/img/17nfp520wu29sjpg/original.jpg (visited on 07/02/2014).

[58] ubuntuusers. *LIRC*. URL: http://wiki.ubuntuusers.de/Lirc (visited on 06/23/2014).

| UPnP Service Name | Functionality |
|---|---|
| ListPresets | Return: ? = factoryDefaults |
| SelectPreset | ? |
| GetMute | Return: mute on/off |
| SetMute | Return: set mute on/off |
| GetVolume | Return: current volume |
| SetVolume | Return: set current volume |
| GetBrightness | Return: current brightness |
| SetBrightness | ? |
| GetContrast | same same |
| SetContrast | ? |
| GetSharpness | same same |
| SetSharpness | ? |
| GetColorTemperature | same same |
| SetColorTemperature | ? |
| X_UpdateAudioSelection | error |
| X_GetAudioSelection | error |
| X_UpdateVideoSelection | error |
| X_GetVideoSelection | error |

**Table 21:** Available UPnP Services - urn:schemas-upnp-org:service:RenderingControl

| UPnP Service Name | Functionality |
|---|---|
| ListPresets | Return: ? = factoryDefaults |

**Table 22:** Available UPnP Services - urn:schemas-upnp-org:service:ConnectionManager

| UPnP Service Name | Functionality |
|---|---|
| AddSchedule | ? |
| ChangeSchedule | ? |
| CheckPIN | ? |
| DeleteChannelList | ? |
| DeleteChannelListPIN | ? |
| DeleteRecordedItem | ? |
| DeleteSchedule | ? |
| EditChannelNumber | ? |
| EditSourceName | ? |
| EnforceAKE | not supported |
| GetAllProgramInformationURL | ? |
| GetAvailableActions | Return: List with all available actions |
| GetAVOffStatus | not supported |
| GetBannerInformation | not supported |
| GetBDVideoID | not supported |
| GetChannelListURL | Return: http://192.168.0.103:9090/BinaryBlob/2/ChannelList.dat -> URL to file |
| GetChannelLockInformation | ? |
| GetCurrentBrowserMode | ? |
| GetCurrentBrowserURL | ? |
| GetCurrentContentRecognition | Return: ok, channel name, Program Title |
| GetCurrentExternalSource | Return: ok, current external resource (TV, HDMI, Scart,...) |
| GetCurrentMainTVChannel | ok, current channel in xml notation ( confusing) |
| GetCurrentProgramInformationURLReturn | ok, http://192.168.0.103:9090/BinaryBlob/4/CurrentProgInfo.dat -> URL to file |
| GetDetailChannelInformation | ? |
| GetDetailProgramInformation | ? |
| GetDTVInformation | Return: ok, xml with dtv informations |
| GetHTSAllSpeakerDistance | not supported |
| GetHTSAllSpeakerLevel | not supported |
| GetHTSSoundEffect | not supported |
| GetHTSSpeakerConfig | not supported |
| GetMBRDeviceList | Return: ok, xml with MBR Device List |
| GetMBRDongleStatus | Return: ok, MBR Dongle Status (enabled, disabled, ?) |
| GetRecordChannel | ? |
| GetRegionalVariantList | ? |
| GetScheduleListURL | Return: ok, http://192.168.0.103:9090/BinaryBlob/1/ScheduleList.dat -> URL to Schedule List |
| GetSourceList | Return: ok, List with all output source interfaces (HDMI1, HDMI2, ...) |
| ModifyChannelName | ? |
| ModifyFavoriteChannel | ? |
| layRecordedItem | ? |
| ReorderSatelliteChannel | not supported |
| RunBrowser | run Browser with provided URL |
| SendBrowserCommand | ? |
| SendMBRIRKey | ? |
| SetAntennaMode | set Anntenna to Mode 0,1,2,... |
| SetAVOff | not supported |
| SetChannelLock | ? |
| SetHTSAllSpeakerDistance | not supported |
| SetHTSAllSpeakerLevel | not supported |
| SetHTSSoundEffect | not supported |
| SetMainTVChannel | ? |
| SetMainTVChannelPIN | ? |
| SetMainTVSource | ? |
| SetRecordDuration | ? |
| SetRegionalVariant | ? |
| StartCloneView | not supported |
| StartExtSourceView | ? |
| StartInstantRecording | ? |
| StartSecondTVView | not supported |
| StopBrowser | ? |
| StopRecord | ? |
| StopView | ? |

**Table 23:** Available UPnP Services - urn:samsung.com:service:MainTVAgent2

| id | requirementTable | requirementValue | sourceTable | sourceValue | event Level | eventDescription | device |
|---|---|---|---|---|---|---|---|
| 1 | mined_data_events _level_1 | device_friendly_name= [TV]UE40ES6300 | mined_data_device _on_off_status | device_friendly_name= [TV]UE40ES6300, status=device turned on, event_date=<date> | 2 | <date> Samsung Smart TV UE40ES6300: turned on | [TV] UE40ES6300 |
| 2 | mined_data_events _level_1 | device_friendly_name= [TV]UE40ES6300 | mined_data_device _on_off_status | device_friendly_name= [TV]UE40ES6300, status=device turned off, event_date=<date> | 2 | <date> Samsung Smart TV UE40ES6300: turned off | [TV] UE40ES6300 |
| 3 | mined_data_events _level_1 | device_friendly_name= [TV]UE40ES6300 | raw_data_upnp _channel | channel_name= <channelName>, occurance=<date> | 2 | <date> Channel: <channelName> selected | [TV] UE40ES6300 |
| 4 | mined_data_events _level_1 | device_friendly_name= [TV]UE40ES6300 | raw_data_upnp _external_source | external_source= <externalSourceName>, occurance=<date> | 2 | <date> External Source: <externalSourceName> selected | [TV] UE40ES6300 |
| 5 | mined_data_events _level_2 | any | raw_data_upnp _channel | volume=<volume>, occurance=<date> | 3 | <date> Volume: <volume> | [TV] UE40ES6300 |
| 6 | mined_data_events _level_2 | any | raw_data_upnp _channel | mute=<mute>, occurance=<date> | 3 | <date> Mute: <mute> | [TV] UE40ES6300 |
| 7 | mined_data_events _level_2 | any | raw_data_network _traffic | category=hbbtv, message_type= HTTP Request, occurance=<date> | 3 | <date> HbbTV Request | [TV] UE40ES6300 |
| 8 | mined_data_events _level_2 | any | raw_data_network _traffic | category=hbbtv, message_type= HTTP Response, occurance=<date> | 3 | <date> HbbTV Respose | [TV] UE40ES6300 |
| 9 | mined_data_events _level_2 | any | raw_data_infrared | key=<key>, occurance=<date> | 3 | <date> Key: <key> pressed | [TV] UE40ES6300 |
| 10 | mined_data_events _level_2 | Channel: <channel-Name> | raw_data_upnp _channel | program_title= <programName>, channel_name= <channelName>, occurance=<date> | 3 | <date> Program watched: <channelName> - <programName> | [TV] UE40ES6300 |
| 11 | mined_data_events _level_2 | External Source: Browser | raw_data_upnp _browser | browser_mode=Tab, browser_url=<url>, occurance=<date> | 3 | <date> URL requested: <url> | [TV] UE40ES6300 |

**Table 24:** Rule Structure of Implemented Prototype