Submitted by
Stefan Reinthaler, BSc.

Submitted at
Institute for Network and Security

Thesis Supervisor
Assoc.-Prof. Mag. Dipl.-Ing. Dr. Michael Sonntag

July 2018

A disaster and crisis alerting approach employing home automation systems

Master's Thesis
to confer the academic degree of

Diplom-Ingenieur
in the Master's Program

Computer Science

JOHANNES KEPLER UNIVERSITY LINZ
Altenberger Str. 69
4040 Linz, Austria
www.jku.at
DVR 0093696

ABSTRACT

This thesis deals with the conception and development of a modern disaster and alerting approach employing home automation systems. This approach illustrates how home automation technologies (smart homes) can be integrated in disaster alerting process starting from event notification up to possibility for automated risk mitigation for self-protection. Besides an insight into the disaster and crisis management and traditional and current alarming methods, structure, functionality and capabilities of smart home integration are highlighted. The goal is to examine technical characteristics that should be considered for the design and technical implementation of a scalable and fail-safe emergency alarm system. These aspects such as general requirements of the system, components, interfaces, communication infrastructure and techniques are highlighted and compared. The result is a conceptual design and implementation of a prototype of a disaster alerting system employing home automation systems.

KURZFASSUNG

Die vorliegende Masterarbeit befasst sich mit der Konzeption und Entwicklung eines modernen Ansatzes im Bereich Katastrophen- und Notfallalarmierung unter Einsatz von Hausautomatisierungstechnologien. Es soll beantwortet werden, inwiefern Hausautomatisierungstechnologien (Smart Home) bei der Katastrophenalarmierung, von der Benachrichtigung über ein Ereignis, bis hin zum Ergreifen von Selbstschutzmaßnahmen zur Risikominimierung, unterstützen und eingebunden werden können. Neben einem Einblick in das Katastrophen– und Krisenmanagement und einer Gegenüberstellung aktueller Alarmierungsmethoden, erfolgt die nähere Betrachtung von Aufbau, Funktionsweise und Möglichkeiten von Smart-Homes. Vorrangig werden aber technische Aspekte untersucht, die für das Design und der technischen Umsetzung eines skalierbaren und ausfallsicheren Notfallalarmsystems in Betracht gezogen werden sollten. Diese Aspekte betreffen allgemeine Anforderungen an das System, Anbindungsschnittstellen, Infrastruktur und Kommunikationsmöglichkeiten, welche näher beleuchtet und gegenübergestellt werden. Das Ergebnis dieser Untersuchung ist eine konzeptionelle Beschreibung und die prototypische Umsetzung eines Katastrophen- und Notfallalarmierungssystems mit Hilfe von Smart-Homes.

Table of Contents

List of Figures

List of Tables

Listings

Table of Formulas

A disaster and crisis alerting approach employing home automation systems

1. Introduction

As the history has shown, many disasters and crisis have happened, which were not really predictable for human beings, even supported by modern technology systems. Reasons for the occurrence of such incidents are natural origins, technical failures or triggered due to a misbehavior from humans. Such critical events can influence a small region, nationalities or can even have an international scope. Whatever an unpredictable disaster causes, it is always necessary to alert affected people in an effective and time-efficient way. Especially for highly life-threatening situations, it is required to react appropriately to mitigate the risk. Hence, efficient disaster and crisis management and disaster and especially crisis communication is an important. Disaster management enables organizations and governments to identify risks and plan suitable measures to handle incidents.

As stated by United Nations Office for Disaster Risk Reduction [1, p. 4], national authorities are responsible to perform disaster management. They have to provide general information about potentially harmful scenarios, the warning mechanisms and recommend risk mitigation actions in case of emergencies. For instance, a disastrous situation can arise in case of a chemistry accident in an industrial complex, assuming that hazardous chemicals were released by an explosion, which are harmful to the surrounding population. While first responders (e.g. fire brigade) trying to get the situation under control, in order to reduce the risk, everyone is personally responsible to protect himself from the consequences in the end. Therefore, it is essential for affected individuals to accordingly respond and perform a predefined set of actions, based upon the type of incident to deal with a threatening situation. For example, one recommended measurement for self-protection is to close all open windows in the building and disable the ventilation, in order to avoid contamination because of a toxic or radioactive cloud. Especially, the time of recognizing a warning notification, up to performing the proper mitigation activities, can be a critical factor in life-threatening situations.

In a historical view, a proven and widespread method for warning people in the past (and still today) is usage of classic sirens to inform affected regions about an incident. Sirens are very loud and it is possible to reach many people with a single signal. Especially the simple set-up and the fact that people can be warned even if the infrastructure is damaged, makes them still indispensable. Beside of recognizing a danger and dissemination of warnings by a national authority, the perception of warnings by the affected people is essential. The perception of warnings presupposes that affected people can understand and interpret the alert information to respond accordingly. In other words, people are facing the problem that they have to know different alarms signals, their significance and the associated risk mitigating actions to deal with an upcoming threatening situation.

With increasing dissemination of modern technologies, it makes sense to think about additional disaster warning and alarming approaches using these new communication channels. At this time, new alerting and warning approaches show up on this field of research. The focus is especially on new warning methods that are assisted by new technologies for increased determined communication. The new communication types are ranging from SMS, E-Mail up to mobile phone applications. As an example, the German Fraunhofer-Institute has developed an alerting system by using such modern technologies.

With this warning system it is possible to alert people via a mobile phone application. The system not only sends information about what has happened, it is also able to give textual instructions, how to behave in case of emergency.

However, with the introduction and the distribution of home automation systems, a new opportunity is created to take another step in this development process. The functionality of automating tasks within a building provided by home automation systems, regarding risk mitigation activities within emergency situations, offers a new ability for developing an extended disaster alerting and response approach. So it is obvious to investigate and develop a new approach to extent the disaster and crisis alerting considering home automation systems.

In general, the term home automation can be seen as a synonym for building automation. Overall, building automation systems are able to perform tasks automatically, based on predefined events. So all elements (e.g. sensors, actuators or other technical units) that are connected within a building can be monitored and controlled by this system. It is often used to control lights, the heating and air condition system, windows and entry points within a building from a central point. The classic building automation systems are widely spread especially in industry and enterprises, but due to convenience and efficient handling of all day tasks, such systems are increasingly employed in the private sector.

1.1. Objectives and Expected Result

The main goal of this is to design a holistic disaster alerting approach employing home automation systems that address the communication and response challenges in early warning. In other words, the envisaged concept should demonstrate, how home automation systems can technically be integrated in a disaster alerting process, in order to alert people about upcoming incidents and to provide a new possibilities for affected people to perform risk mitigation actions leveraging smart home capabilities.

The following figure illustrate an abstract view of the envisioned concept:

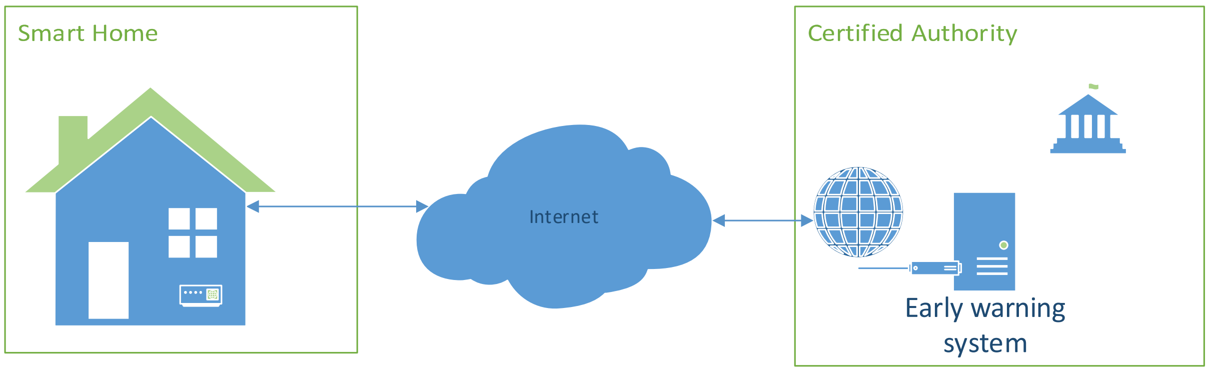


Figure 1: Main components of the disaster alerting approach

The basic idea is to setup a reliable information system that enables the communication between disaster alerting services and smart buildings. The realization of such a system requires the interaction of several components. On the one hand, a source of information is required that provides information about current disasters or incidents. This can be an emergency warning service hosted by a national authority, which is accessible via an internet connection. The information service provides an interface to answer requests sent by the smart homes. One the other hand, it requires a middleware or and endpoint to enable home automation systems to request this source of information and perform further tasks.

This brief description of basic components and their interaction should give an idea, how smart homes can be equipped with disaster alerting and response capabilities. Besides the technical requirements, the main challenge will be to design a reliable and secure communication system and to give an example in terms of a prototype to integrate the bunch of different home automation systems. The methodical approach to achieve these objectives is introduced in the next section.

1.2. Methodical Approach & Structure

The focus of this work is on designing and implementing an enhanced approach by employing home automation systems in the disaster alerting process. The concept is divided in two major parts. The first part focuses on the theoretical aspects of disaster and crisis alerting approaches and state of the art of the current warning approaches to identify elementary functionalities and techniques. In order to fit the needs of the envisioned disaster and crisis alerting system, a screening of the literature and best practices is required to ensure a high level of quality. This necessitates requirements engineering and system specification. Because of the fact that the system is designed to interact over the public internet with private households, privacy, security and reliability also have to be in focus. All the information obtained during the theoretical work is the basis to develop a concept and a prototype, introduced in the second part.

Besides the theoretical findings, these research questions should guide during the technical specification and design phase of the concept and prototype:

- What functional and non-functional requirements are mandatory for a disaster alerting system?
- Which infrastructure and communication design is useful regarding reliability, scalability and performance?

In order to develop a disaster and crisis alerting system employing home automation systems, it will be necessary to identify important aspects in disaster and crisis management. The next section will start giving insights about the concept of disaster and crisis management in context with civil protection.

1.3. Disaster and Crisis Management

According to Drabek and Hoetmer [2], disaster and crisis management focuses on creating plans to reduce the effects of disasters, in order to prevent damage to assets, human mortality, and lost revenue. The following sections provide a general overview about disaster and crisis management, the characteristics of disasters and crisis, insights into the life cycle approach and the relevance of effective communication in context with civil protection.

Definition and Characteristics of Disasters and Crisis

The understanding of crises and disasters are not necessarily congruent with the parlance of the population, the understanding of private enterprises or in a governmental perspective. Even the escalation model of a company, for example, distinguish between the normal case, emergency, crisis and disaster. So, the following definitions and characteristics are trying to cover the overall view in disaster and crisis management.

A disaster or crisis is usually defined in terms of two conceptual elements [3, p. 6]:

- Some occurrence that risks or harms the life or health of people or animals, the environment, significant property or the vital supplies of the population.
- Secondly, the defense and fight against this happening requires uniform management by a competent authority, especially for civil protection.

In addition, disasters and crises can have various causes and vary considerably in each effect. The following events can trigger disasters and crisis [3, p. 6]:

- Natural disasters: extreme weather conditions (e.g. a storms, heavy rainfall, floods, heat wave), forest fires, earthquakes and epidemics.
- Technical or human failure: System failure, negligence and accidents.

Furthermore, each crisis has its own characteristics or can be seen as unique. Nevertheless, there are patterns that occur regularly, which helps for preparation and analysis. So, disasters and crises can be captured in various dimensions:

- A real event with a concrete danger or damage effect to sensitive entities (persons, property, nature etc.).
- The way of crisis management by the competent authorities and companies (including pre- and post-processing).
- The perception of crisis management by the stakeholders and the public.

In order to handle disasters and crisis successfully, a comprehensive disaster and crisis management has to take place. The method generally used by national authorities to manage disasters and crises, is presented in the next section.

1.3.1. Disaster and Crisis Management Life Cycle

The disaster and crisis management life cycle can be seen as a framework for handling exceptional events, which are not part of daily business. Hence, it is a comprehensive method and can be applied in different levels, for example supporting authorities, organizations or enterprises to protect the environment, people and goods.

According to the Bundesministerium des Innern [3], disaster and crisis management in a contemporary understanding can be seen as an ongoing process. It presents itself in the form of a life cycle, which consists of four phases, as seen in Figure 2:

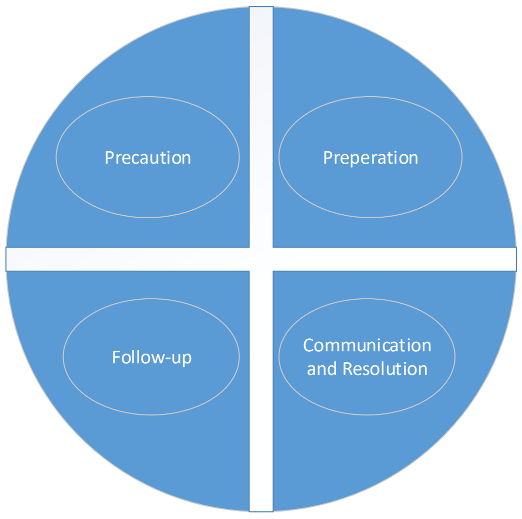


Figure 2: Disaster management life cycle, related to [4, p. 17]

The precaution and preparation phase takes place in the normal state. At this state, crisis management has mainly anticipative character and is used to prevent crisis. Basically, potential crises are identified in order to set preventive measures as a first step. The next step is to identify measures to protect against potentially occurring crises. This is already a kind of future-oriented compensation. The preparation phase also includes the implementation of a risk analysis in the context of risk management, the establishment of early warning systems and the definition of processes for possible emergencies, e.g. alerting processes and plans [3, p. 7].

For the planning of crisis management, the above-mentioned potential level of a latent and acute crisis has to be considered. In the latent phase of the crisis, early warning systems report risks to people and goods. In this phase, both measures to prevent as well as measures in case of an escalation to an acute crisis have to be taken. In practice this means that existing concepts have to be adapted to the expected crisis scenario, because every crisis has its own challenges. So, depending on the specific emergency plan, it could be already necessary to alert affected people about the current situation. Furthermore, if a disaster or crisis occurs, the stage changes from latent to acute. At this time, predefined actions are required including an immediate communication, even if the information of the current situation have not been fully collected. This means that an immediate alert has to be given to emergency responders on site, for example ambulance services, fire brigades, police, emergency teams in companies and others. A central point in this initial phase of a crisis, is the immediate coordination and mutual information between relevant authorities and companies, the media, the general public and also experts.

The follow-up phase within a crises process is an important component and basically starts already during the communication and resolution phase of handling an emergency. A primary task for example is the appropriate documentation of all actions within the handling process. This makes it possible to review the process and evaluate them. This step can result in a report, which covers new aspects of the handling process that can be considered in current emergency plans [3, p. 8].

1.3.2. Relevance of effective Communication

Risk and crisis communication is an elementary part of successful disaster and crisis management. Risk and crisis communication are often mentioned together, so that the impression can arise, both terms describe more or less the same topic [3, p. 11]. Indeed risk and crisis communication are closely connected with each other, because risk communication is the basis for successful communication in crisis scenarios. Nevertheless, risk and crisis communication also differ in very important points.

A major difference between risk and crisis communication is given in the temporal dimension. The goal of risk communication is especially the prevention and the preparation for threats and risks. Risk communication should permanently develop and maintain a relationship of trust with the target groups. Therefore, risk communication aims to sensitize people of existing dangers and risks within their environment, in order to educate them about how to deal with potential situations, by providing recommendations for preventive measures. It is also mentioned, that the management of large-scale emergencies and disasters, with increasing size and duration, requires a corresponding participation of the population. In order to empower citizens to protect themselves by performing self-protection measures, the potential scale and consequences have to be communicated prior to the occurrence of an event for a long time. Only preventative communication of self-protection measures can ensure, that affected people have the appropriate knowledge what to do.

In contrast, crisis communication is characterized more by short-term, time-limited action that substantially prevents acute threats or limits damage already incurred to lead back to a normal state as soon as possible. According to the Bundesministerium des Innern, crisis communication is defined as…

"…exchange of information and opinions during a crisis to prevent or limit damage…", [3, p. 11]

In other words, effective crisis communication should enable affected people to prevent or limit the imminent damage to their life and property after the occurrence of a disasters, and subsequently with the aim to return to a normal state as soon as possible. The target groups can vary depending on the type of disaster or crisis. In general, disaster and crisis management (within organizations or enterprises) target groups can include customers, suppliers, employees, the media, other agencies or authorities, scientific institutions, companies, associations, interest groups. In a view of civil protection, the target group of crisis communication process may be primarily civilians. However, all these target groups need to be informed. Hence, all alerts or messages and published information should be kept in a reasonably factual and clear written style.

Finally, disaster and crisis management and effective risk and crisis communication as mentioned above, have a high impact of the success of handling unforeseen situations. The concept and four phases of the disaster management life cycle introduced, can also be found in national disaster and crisis management in context with civil protection.

1.3.3. Civil Protection in a Governmental View

The protection of people is the primary goal of the civil protection. Civil protection calls for an interaction of all forces to deal with the hazards and disaster risks. This involves authorities and emergency services and require the assistance of the population as well. Furthermore, civil protection are all measures to protect the population and the public in the event of crises.

In a governmental view [5], it is specially emphasized that the self-protection is an essential part of the civil defense and disaster protection. In addition to the responsibility of the official authorities (organized civil protection), individuals are urged to equally contribute to enable successful disaster protection. Moreover, the population itself is called upon to take reasonable preventive and response operations in context with self and mutual aid. This implies that the population must actively participate in the disaster handling and recovery process and to be well prepared in case of emergency.

This aspect requires a sufficient and rapid flow of information from and to the authorities. For this purpose, a system of emergency warning centers (EWC) have been established at a national and regional level, for example to Federal Alarm Center (FAC) in Austria, where all information arrives about national incidents or abroad. A key concern of emergency warning centers is the rapid warning and alerting of the population in a disaster or crisis situation. Moreover, these are also point of contact for relief and rescue organizations such as firefighters, Red Cross and also the contact point for the neighboring states, the European Union, the NATO Partnership for peace and the United Nations [5].

However, EWC employ various early warning systems (EWS) to warn the population as effectively as possible. The aim is to inform the public through multiple channels in order to reach a large part of the population and to guarantee that the warnings are perceived. At this point, we take a closer look at early warning in the next section.

1.3.4. People-Centered Early Warning

The goals of people-centered early warning is to enable people and communities threatened by disasters to respond in adequate time and in a sufficient way so to reduce the probability of harm, loss of life, damage to property and environment. Early warning (EW), as defined by Grasso, Singh and Pathak [6] is…

"…the provision of timely and effective information, through identified institutions, that allows individuals exposed to a hazard to take action to avoid or reduce their risk and prepare for effective response".

In general, early warning require the integration of four key elements: Risk knowledge, monitoring and warning service, dissemination and communication, and response capabilities [1, p. 2].

- Risk knowledge: Risk management and assessment help to prioritize mitigation strategies and construct early warning systems.
- Monitoring and warning service: Monitoring and predicting systems enables to evaluate the potential risk.
- Dissemination and Communication: The dissemination of warning messages requires a reliable communication scheme to reach affected locations and to alert local and regional governmental agencies. In addition, messages have to be in a consistent and understandable form.
- Response: A central aspects of early warning includes efficient organization, guidance and action plans, public awareness and education.

This requirements of early warning are closely connected to the disaster and crisis management characteristics introduced in section 1.3.1.

Derived from these characteristics, key aspects could be identified that are essential to effective self-protection:

- **Awareness**: Special attention of emergency warning mechanisms for people-centered early warning is on the wake-up effect, which is the ability to make the population aware of disasters.

- **Information**: The second important characteristic of efficient early warning was identified as informational aspect (Information). Context sensitive and timely information about disasters, for example what has happened, where, and which activities can be taken to reduce the risk of harmful situations are crucial to ensure the best possible protection for affected people.

- **Reaction**: The third aspect identified for an improved warning process is the possibility to enhance the reaction capabilities (Reaction). Only time-efficient and proactively performed activities are the best way for protection.

In the following, the state of the art of alarming systems is given, to give an overview of currently used warning mechanisms. The quality and suitability of disaster alerting systems can be described and measured on the basis of the characteristics above. The subsequent comparison should highlight the potential of employing home automation systems.

1.4. State-of-the-Art of Disaster Warning Mechanisms

The different alerting approaches are investigated in respect to the awareness, information and response aspect. The first part focus on more traditional alerting mechanisms, whereas the second part give attention to methods using new technologies.

- **Sirens:** As already mentioned, one of the simplest, most commonly used approaches to alert the public about dangerous situations, is the use of a comprehensive central controlled siren network. Due to the high range of the warning signal and the high wake-up effect, this method has been proven, since no additional equipment is needed for humans to get note of alarms. So, alarms can be locally sent to the public, depending on the type and impact of incident. This alerting method is quite effective in respect to the wake-up effect, but is strongly limited transmitting information over this channel. Basically, it is the job of the individual to interpret the acoustic signals correctly, in order to associate the signals with the right mitigation actions. Therefore, individuals are responsible to actively inform themselves about possible dangers, the meaning of alarm signals and specific risk mitigation actions. The reason for this is the very low informational content that can be transmitted using this alarm channel. Hence, the informational aspect cannot be actively supported. But in respect to the self-protection idea, fast and correct information is an essential part of a comprehensive alert mechanism. To achieve this, additional warning mechanisms are necessary.

- **TV and Radio:** Another classic method to inform the population of current emergencies is done using entertainment media, e.g. TV and radio. With the use of radio and TV, current status reports can be distributed in an easy and simple way. The ability to track the current situation is the major advantage using this source. This way of information seems to be well suited for supporting the informational aspect, but only if people are already aware of an incident. Although, there are some additional things to note regarding the informational content. In principal, TV and radio programs are broadcasting media and can be consumed independent of the national region. As a result, current consumers of a media channels are urged to assess their current situation based on the context of the provided incident information, even if they are not affected. So, the dissemination of incident information cannot be really targeted for a specific region. It is basically the same with sirens. From an alerting perspective, the wake-up effect is missing. That means that these devices have be running in order to catch notifications about current incidents. It also has to be noted, that the younger generation rather prefer TV and music streaming services or internet-based radios, than listening to traditional radios. This can be seen as weakness, besides the fact that such a device is currently available at all.

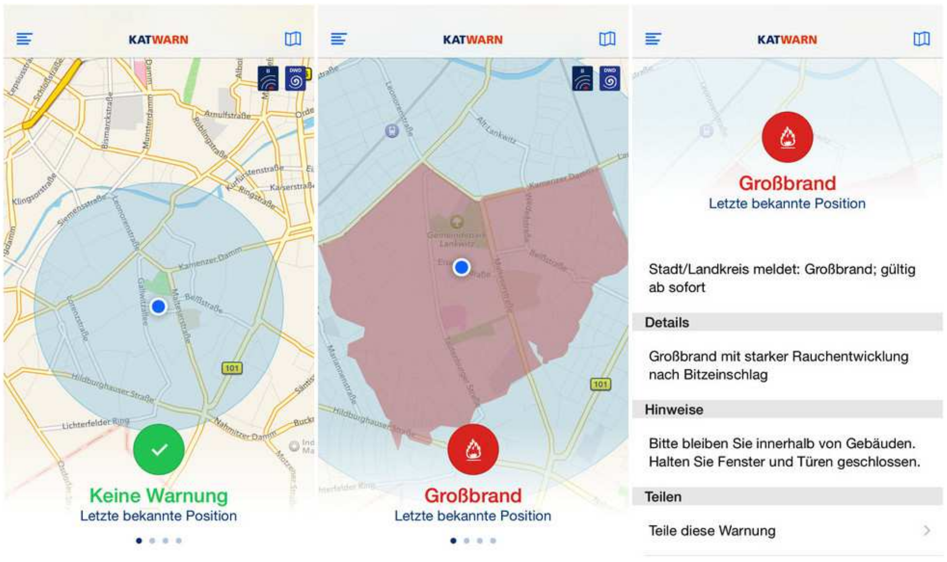
- **SMS – Short Message Service:** Due to the progress of technology, new alerting mechanisms show up on this field. Nowadays, many people already own mobile phones. With mobile phones, individuals are able to cover a large part of their information and communication needs.

Therefore, they have become an indispensable part of our world. Due to the fact that smart phones are widely used, almost continuously online and perform information and communication tasks in daily business, they are well suited to be part of a modern emergency warning system. Besides the traditional warning methods, as sirens or TV and Radio, delivery of emergency messages via the Short Messaging Service is an increasingly accepted and used method. Affected people can be easily notified. This method also supports the desired wake-up effect and informational aspect. But it is limited to short text based messages only.

Besides the traditional approaches of alerting, additional methods are developed that increasingly use new technologies to extend and improve the alerting process. In the following, several new methods are presented, which have peculiarities.

- **Disaster Information Systems:** Another approach for monitoring of disasters is the application of disaster information systems. ICT enables governmental organizations building a network of monitoring and alerting systems. Such a system as the Global Disaster Alert and Coordination System (GDACS) [7], which is a coordinated web-based platform, providing alerts and impact estimations for earthquakes, tsunamis, floods or cyclones. Information about current disasters can be seen on a global map. It also provides a personalized alerting functionality via SMS, E-Mail, RSS feeds or mobile phone applications. In addition, GDACS uses the standardized Common Alert Protocol (CAP) [8] to provide an interface for data and information exchange with other systems. There are many other disaster information systems following the same principle. Meteoalarm [9] is a web-based service for alerting European countries about extreme weather situations. The Integrated Public Alert and Warning System (IPAWS) [10] is an emergency population warning system in the United States. These systems focus especially on natural disaster events. All these disaster information systems can be sources for retrieving high quality information of dangerous situations. So, disaster information systems are well suited to support the awareness and informational aspects on a high level.

- **Mobile Phones Applications:** As research has shown, a variety of mobile services exists, for a variety of warning use cases. The KATWARN [11] system is originally a German warning service for mobile phones and is also introduced in Austria today. In case of disasters like wildfires, bomb finds or hurricanes, responsible authorities as the Federal Emergency Management Agency, fire control centers or the national weather service, can send warning information directly and spatially related to the mobile phones of the concerned citizens (location based). With the location-based alerting service, certified authorities are able to notify users about current incidents and send behavioral information. In case of emergency, national authorities are able to select affected users and regions using the ZIP-Code, generate specific warning messages and sending alarm or all-clear messages. For the use of the application, no registration or authentication is required. The service uses the geographical information for receiving appropriate information. The user is also able to share alarm messages within social media platforms (e.g. twitter) to notify friends and family.



Stadt/Landkreis meldet: Großbrand; gültig ab sofort

Details

Großbrand mit starker Rauchentwicklung nach Bitzeinschlag

Hinweise

Bitte bleiben Sie innerhalb von Gebäuden. Halten Sie Fenster und Türen geschlossen.

Teilen

Teile diese Warnung

Figure 3: KATWARN example [11]

In addition it is possible to show the received information in a map for better visualization. Furthermore, the KATWARN warning system can also be used without the mobile application installed. Users can obtain warnings via an SMS or E-Mail, through registering their location by sending a message with the specific ZIP-Code. KATWARN seems to be a modern emergency warning system, in addition to traditional loudspeaker announcements or sirens. However, the stability of this communication system strongly depends on the availability of mobile network and scalability of the server providing this service. So, it happened that during an emergency in August 2016 that the service was not reachable for a particular time and warning messages were sent with a delay of several hours.

BIWAPP [12] is another modern smartphone app that informs and warn the German public in case of emergency. It follows the same principle as the KATWARN system. Civil protection authorities and their control centers can send important information and warnings about incidents and threats to the app and thus reach citizens in an effective and direct way. Unlike KATWARN, this app offers a bit more functionalities and individual adjustment. Places and themes (nature of the hazard) can be individually selected, in relation of receiving information and warning messages. Another feature provided, is the ability to make emergency calls out of the app, which allows to inform the police or fire brigades about the current situation. In principle, this is not really a new feature, because emergency call can be made without the need of an application. At this point it has to be noted, that emergency calls from mobile phones can no longer (since 2017) be made without an activated SIM card in Germany. The reason for this limitation is to prevent the misuse of emergency numbers.

- **Digital Radio with EWF:** The Fraunhofer Institute Fokus, in cooperation with NOXON Vertriebsgesellschaft mbH, TMT GmbH & Co. KG und Bayern Digital Radio GmbH, developed a new emergency warning approach, on implementing the wake-up effect in digital radios. The mentioned feature is called “Emergency Warning Functionality” (EWF) [13] and is based on the Digital Audio Broadcasting Plus Standard (DAB+).

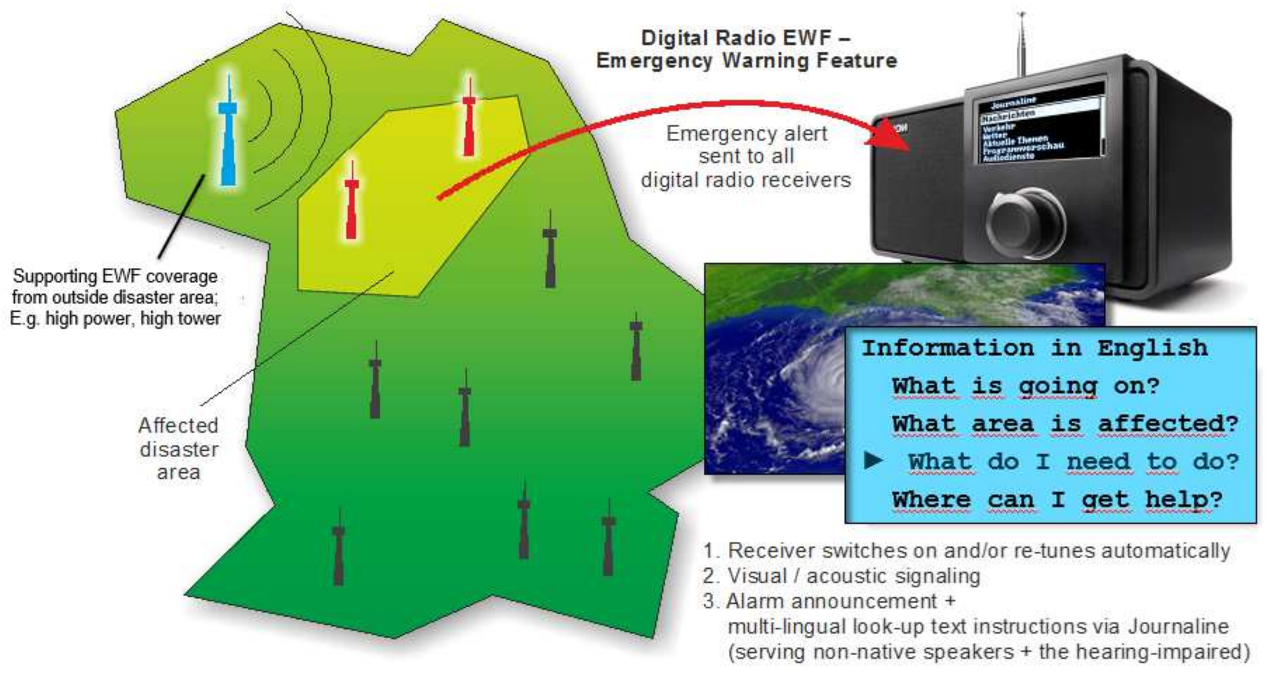

Digital Radio EWF – Emergency Warning Feature
Emergency alert sent to all digital radio receivers
Supporting EWF coverage from outside disaster area; E.g. high power, high tower
Affected disaster area
Information in English
What is going on?
What area is affected?
What do I need to do?
Where can I get help?
1. Receiver switches on and/or re-tunes automatically
2. Visual / acoustic signaling
3. Alarm announcement + multi-lingual look-up text instructions via Journaline (serving non-native speakers + the hearing-impaired)

Figure 4: Digital Radio EWF - Architecture [13]

So, in an emergency situation, an alarm is sent by an authorized emergency center, all digital radios pick up the alarm signal and switch to the emergency broadcast, even if the device is in stand-by mode. To reinforce the wake-up effect, additionally conspicuous signals as visible flashing of the display or volume increase are used. In addition to the audio announcement, an emergency message appears on the screen of the receiving device. The text based content can include general information and instructions, text-headlines or images. Furthermore, the warning messages can contain detailed information on the current situation, competent local contacts, the affected area or the correct behavior in an emergency.

- **Building automation:** Whereas the approaches above are limited to awareness and informational aspects of disaster alerting, first extended approaches show up on the field of research, considering the reaction time. A first prototype of an advanced disaster alert system was introduced by Lin et al. [14], which take care of the automation of tasks, for increasing the reaction time. The prototype called Active Disaster Response System (ADRS) is designed to alarm people and perform emergency tasks within buildings when an earthquake happens. The system can interpret messages based on the Common Alerting Protocol (CAP) [8], which is a standard format for exchanging disaster alert messages. These messages, published by national authorities, are parsed and actuate embedded controllers for executing tasks, for example opening doors, stop the elevator or turn gas pipelines off. In addition the system can provide a temporary network to allow posting emergency messages from victims via mobile phones, which are trapped inside a building. This information, for example their exact location, can help first responders to improve the salvation of the victims.

Comparison of alert mechanisms

The mentioned alarm mechanisms focuses on awareness and information aspects, because of technical limitation. But for an enhanced approach also reaction capabilities are desired. For the evaluation against these alerting mechanisms, different measures are used to express the level of support: low, moderate and high. High means that a certain aspect can be fully supported. Moderate indicates that the aspect can be supported under certain conditions. Low specifies that no or minimal support is given.

The main differences in respect to these aspects, are shown in the comparison matrix below:

Technique	Awareness / Wake-up Effect	Information / Content	Reaction / Automation
Sirens	high	low	low
TV	low	high	low
Radio	moderate	moderate	low
SMS Broadcast	high	moderate	low
Mobile Phone Apps	high	high	low
Digital Radio (EWF)	moderate	high	low
Disaster Information Systems	moderate/high	high	low
Building automation (ADRS)	moderate/high	moderate/high	high

Table 1: Comparison of alert mechanisms

For sure, the goal for a comprehensive and effective people centered warning approach is to support all these aspects in a high level. As shown in the matrix, traditional and also existing alerting approaches, realized with assistance of ICT, differs in the level of supporting the required aspects. Traditional approaches as sirens can generate a high level of awareness, whereas TV and Radio are more in a passive position regarding the wake-up effect. Vice versa, TV and Radio have a high informational level, keeping affected people up-to-date. The informational aspect using sirens is rather low, because this implies that people have to already know what to do depending on the type of alarm signal. In contrast, newer approaches are a bit more in balance and can generate a higher level of both aspects. The informational aspect reaches a high level of all ICT based approaches, due to the fact that different channels and forms can be used to provide informational content. SMS is assessed to be moderate, because to the limited length and textual representation only. In general, the wake-up effect using smart phones for alerting is as high as possible, because smart phones are usually always on and next to their owner. The awareness of digital radio and disaster information systems is assessed as moderate to high level, because digital radio have an wake-up functionality, but this doesn’t work if the device is turned off. Disaster information systems have also a high potential of supporting the awareness aspect, but only in combination with other resources, which have to subscribe to the alarm channel. It seems that best qualification for achieving all the three aspects awareness, information and reaction, provides the combination of DIS, smart phones and building automation. Disaster Information Systems for providing up-to-date information of disasters, smart phones for increased awareness and notification and the integration of building automation systems to automate risk mitigation tasks. These constitution can be the basis of a comprehensive alerting approach.

1.5. Home automation systems

For the development of a disaster and management alerting system employing home automation systems, it is necessary to take a closer look at home automation systems. This understanding, is required in order to design an endpoint interacting with both the home automation system and the warning infrastructure. In this chapter, the central concept of home and building automation systems is outlined.

In general, home automation can be seen as the domestic application of building automation. The main goals for home automation are the optimization of power consumption, cost reductions, enhancing the security and automation of tasks [15]. However, the focus of the targets varies depending on the application type. In contrast to the automation of public buildings and industrial complexes, where energy efficiency, security and cost reduction are the prioritized objectives, is the major focus in the private sector is to increase the quality of living by automating tasks, security and simple remote control of devices.

Many terms exist in the world of home automation. Used terms most in literature and media are smart home, smart living, intelligent home or e-Home. All terms represents nearly the same purpose and can be seen as synonyms in the area of home automation. An overall definition is given by Technopedia [16], describing the main idea of such a system:

> *"A home automation system is a technological solution that enables automating the bulk of electronic, electrical and technology-based tasks within a home. It uses a combination of hardware and software technologies that enable control and management over appliances and devices within a home. A home with an automation system is also known as a smart home. "* [16]

In other words, a smart home or home automation system can be described as a holistic system, which consists of control-units, sensors and actuators linked over a wired or wireless media to enable monitoring and control of connected devices to manage a physical environment. The architecture of such systems is described in the next section.

1.5.1. Architecture of home automation systems

As home automation systems are based on the principle of building automation, it is necessary to discuss the basic architecture and structure of these systems in more detail. This should also provide a better understanding in respect to the interoperability with a disaster alerting system.

Historically, the basic construction of building automation systems are divided in different layers. These three layers are divided into fieldbus, automation and control level. Between the individual layers, interfaces are arranged, which organizes the data and functional transportation arrange on individual data points.

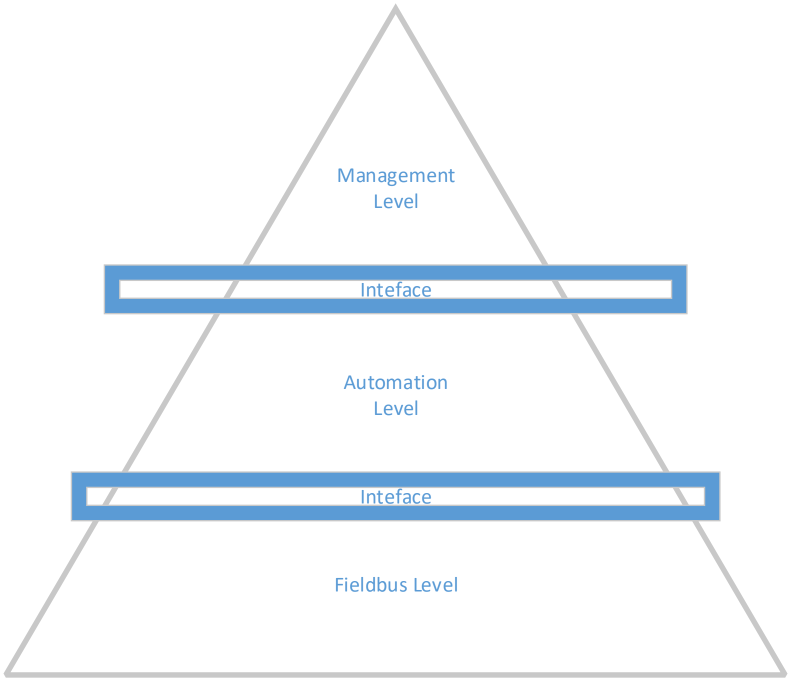


Figure 5: Levels of the automation pyramid [15]

- **Management Level:** The top level of the automation pyramid is the management level. This level deals with the visualization, operation and output of messages and information of the building. From this central position whole buildings can be controlled. In this case, control is not understood in conjunction with automation itself, it is more the ability to control whole processes within a building. Especially this central functionality has high significance in the private sector.

- **Automation Level:** The automation level deals with the components required for building an appropriate logic for controlling, regulating and timing of sensors and actuators. For example, the control of scheduled tasks (calendar or intervals based), the control of states (temperature, brightness, darkness, intensity of radiation, rainfall, air, etc.) or even smart metering. The functions of the automation pyramid are realized by linking modules, logic modules, and controllers.

- **Fieldbus Level:** The fieldbus level is the bus system of the building per se. It consists of sensors and actuators that can communicate with one another within the field bus. Furthermore also gateways are located in this level, which allow the connection between different systems, higher layers or basic facilities of the building systems. In this area, functionalities can already be realized without requiring the automation or management level, like control of lighting, temperature, blinds or air-condition and ventilation.

Sensors and Actuators

Basically, the infrastructure of all home automation systems consists of three major components. Sensors, actuators and a control unit. Sensors and actuators interact with the central control unit, which is responsible for the logic operation. In a technical view, a sensor is a so called detector. A technical component to determine, if some physical or chemical properties have changed in a specific environment. For example, physical values can be the temperature, humidity, pressure, brightness and acceleration or chemical values as ionic strength, electrochemical potential [17]. In a general view, the main task of a sensor is to transform physical or chemical values into electrical signals. In the context of home automation, sensors are essential elements for reporting current states to the controller in order to trigger logical operations. There are many different kinds of sensor components on the market. Most common sensors in home automation systems are motion and occupancy, lighting, temperature, flood and leak, carbon monoxide, proximity, fire sensors. Actuators are exactly the opposite of sensors. Actuators respond to the outgoing command from (electrical signal) from the control unit and converts the signal into a mechanical motion or other physical variables [17].

However, home automation systems differ in terms of the media, the link between different media via gateways, the programmability, the standardization and use of standards, and simply the cost. The following sections should clarify the different implementation types of a home automation system.

Centralized systems

The first criteria of realizing a building automation systems is the distinction for central, decentralized and semi-decentralized systems. A centralized system approach consists of a central controller and connected sensors and actuators. That means that all the communication between sensors and actuators and the required logical operation is executed by the controller (see Figure 6). So, the controller takes care if an event is triggered by a sensor and forwards the predefined instructions to the responsible actuator. The advantage of such a system is that all logical functions can be evaluated by the central controller, because all the states of the connected sensors and actuators are known. Another benefit of this type of architecture is that it is not a prerequisite that a sensor triggers some actions. Activities can also be initialized by some time-based events or simulation functions. On the other hand, centralized systems architectures always have to face one problem. When the central controlling unit fails, for example because of a power outage, system failure or similar, the entire building automation system stops working.

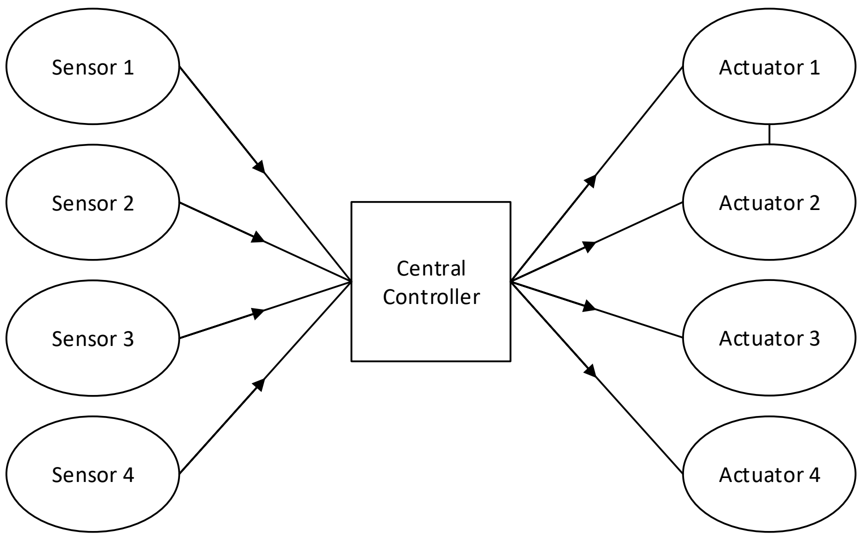


Figure 6: Building Automation - Central System Approach [15]

Most implementations of this kind of system are based on a programmable logic controller (PLC), to manage and control the input and output states of the IO-modules. Another way of implementation is provided by a decentral architecture, described in the next section.

Decentralized systems

According to Aschendorf [15], decentralized systems in building automation are also called “intelligent” systems, due to the fact that every member of the system has to be “intelligent”. In contrast to the centralized system approach, no central control unit exists to take control of the participants. So it is necessary that every participant in the network is equipped with a communication processor, which enables the communication over the network, a processor that handles the functionality of the component and an application processor, which controls the connected sensors and actuators. As a result of that, they are much more expensive. On the other hand, this principle of a stand-alone element can be beneficial, because it is possible to setup preinstalled applications on this components, hence no additional functional programming is required, only the parameterization of the application device with a software toolkit. Such systems are provided for example by KNX/EIB, LON or EnOcean and can be implemented via wired or wireless medium.

The following figure shows the design principle of a decentralized system:

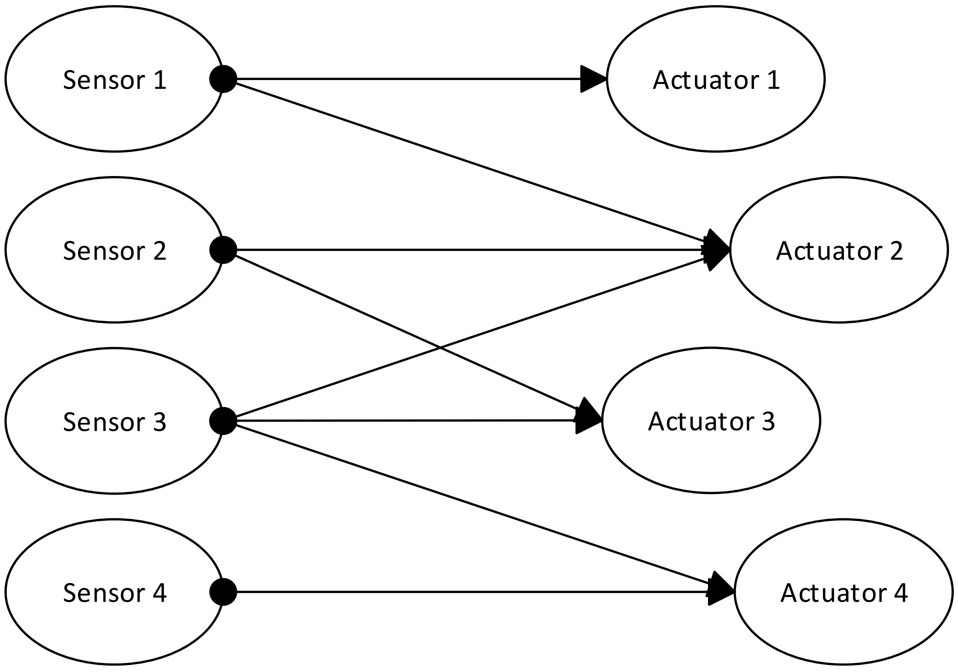


Figure 7: Building Automation - Decentralized Approach [15]

A major problem of such a system design is, if a configuration or behavior of a system has to be changed, the adaptions have to be changed on all involved components. So there is no benefit of a centralized management of the system. On the other hand, the decentralized approach is not that susceptible to full system outages as the centralized concept. In case of a failure only the components are affected where the error occurs, because the system logic is spread over all system components [15]. But this can also cause problems by locating the erroneous components, especially in grown and complex structures.

Semi-Centralized Systems

As described in the above sections, it turned out that both system designs have their advantages and disadvantages. Centralized systems seems to be more static and rigid, because such systems are primary realized with wired medium and the danger of full system outages, if the central controller fails. Decentralized systems are more flexible in contrast to centralized systems, but the configuration changes and maintenance of the systems requires more effort, especially if the system is rare documented. To get rid of the mentioned problematic of both central and decentral systems [15], describe a combined approach of semi-decentralized systems. The clue is to change or adopt centralized systems into semi-decentralized systems by applying more than one central controller. This centralized controllers for example can be connected over simple and stable Ethernet technology, to ensure high performance. A symbolic semi-decentralized architecture is shown in Figure 8.

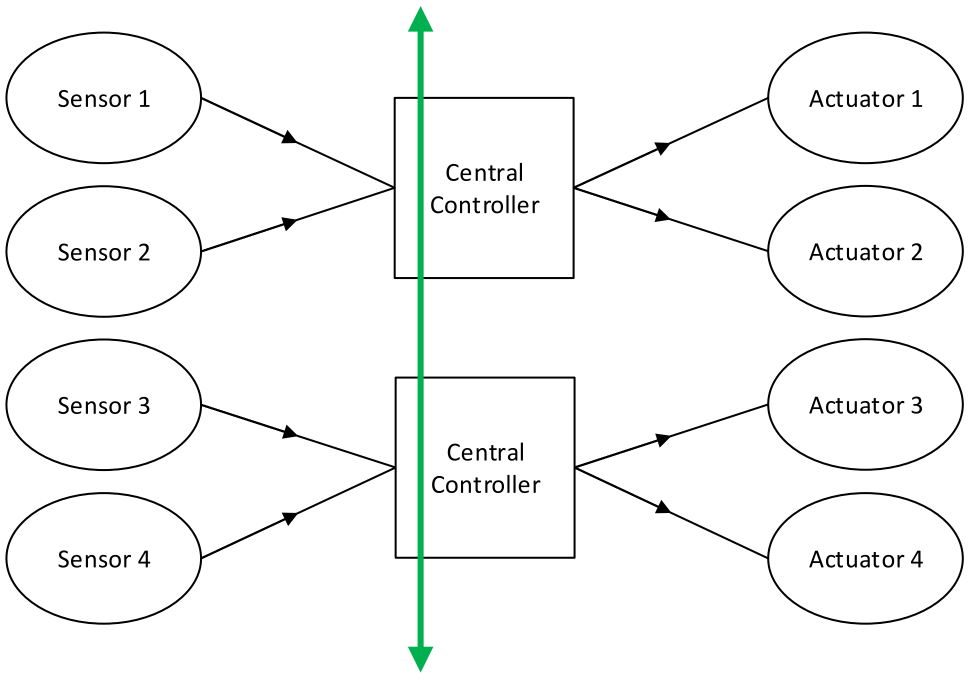


Figure 8: Building Automation - Semi-Decentralized Approach [15]

Finally, this system design makes it possible to reduce the problematic of centralized and decentralized building automation systems. The interconnection and communication of its components can be realized in several ways. The basis is the type of transmission media which are described in the following.

1.5.2. Transmission Media and Communication

Typically employed media in building automation for data transmission and control can be categorized in power supplies for the conventional electrical installation as well as powerline systems, wire-based systems in the form of an additional data line, or radio bus systems. The use of wired-based Ethernet or Wi-Fi increases too.

The following table shows commonly used transmission media for communication in home automation systems:

Technology	Transmission	Frequency	Encryption	Proprietary
ZigBee Pro	Radio	2,4 GHz or 868 MHz	AES-128	No
DigitalSTROM	Power line	-	-	No
Z-Wave	Radio	868 MHz	AES-128	No
EnOcean	Radio	868 MHz	AES-128	No
HomeMatic	Radio/Data link	868 MHz	AES Authentication	Yes
LCN	Power line	3kHz	-	Yes
KNX-RF	Radio	868 MHz	-	No
KNX-PL	Power line	-	-	No
KNX-TP	Data link	-	-	No
WLAN	Radio	2,4 / 5 GHz	WPA, WPA2	No
Bluetooth	Radio	2,4 GHz	AES-128	No
io-homecontrol	Radio	868-870 MHz	AES-128	Yes
DECT ULE	Radio	1880/1900 MHz	AES	No

Table 2: Overview of commonly used transmission standards [18]

Each transmission medium so has its own advantages. Whereas automation systems are largely realized through wired bus systems in industrial buildings or companies, wireless approaches can be found increasingly in the private sector. Brandstetter [19, p. 27] mentioned some basic benefits using wired or wireless approaches. Wired based automation systems have the advantage that transfer rate and range is not limited by structural obstacles, for example using fiber technic, where the transfer rate can reach up to 170Gbps. Also the immunity to electrical and electromagnetic environmental effects is low when using shielded cables in contrast to radio systems. And if required, the transmission capacity can be increased by laying additional cables. For sure the use of wired systems results in higher construction costs. On the other hand, the use of a wireless based system has also benefits. The installation of wireless networks in existing buildings can be easy and affordable, in contrast to the subsequent installation of wires and cables for transmission and control. Another positive effect is, that components connected via a wireless network can be controlled regardless of the location of connection boxes. However, the standalone wireless components are much more expensive than wired components, because they also need a logical unit to fulfill its tasks, which requires additional hardware. In short, establishing a new wireless infrastructure in existing buildings is more affordable than a wired infrastructure, but sensor or actuator components are much more expensive in contrast to wired devices.

Gateways for Interoperability

Due to the fact of gradual development in building automation, the changing technologies and at least many own developments of different manufacturers, no uniform standard exists in building automation that ensures simple interoperability between the bunch of systems, as noted by Aschendorf [15, p. 71]. In order to enable interoperability, building automation system gateways are used to enable the interaction between different de facto standards, media, protocols and other systems. This is done by converting the individual protocols into another suitable format of the target system. Basically, there are different types of gateways. On the one hand software-based gateways exists, which are typically installed on a PC that is directly connected to the automation system. But also hardware-based gateways exists in form of IP-Gateways, as shown in Figure 9. These are tied to the hardware of the bus system or an Ethernet network, which enables the access to the system from any location of the building.

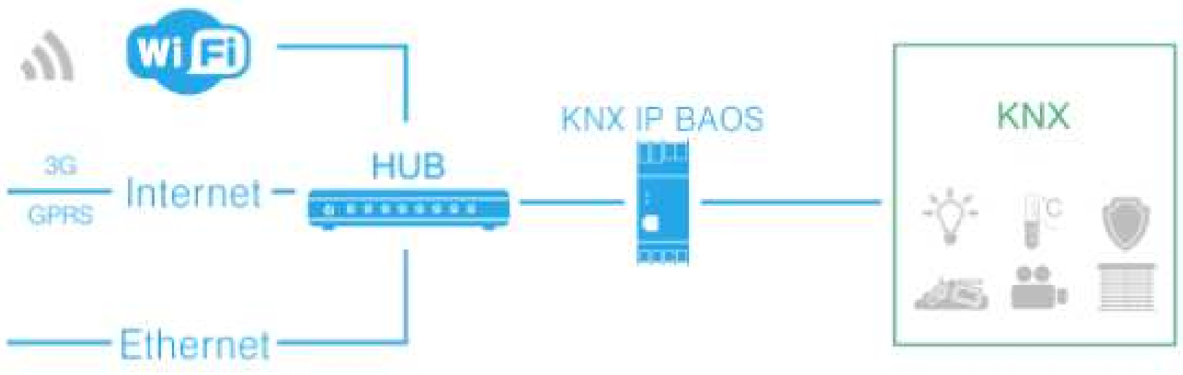


Figure 9: Abstract example of KNX integration [20]

As already mention, gateways enables system engineers to interconnect and access building automation systems over different channels. But also central components for system configuration, management and user reporting will play an essential role in the conception and design of the disaster alerting system employing home automation systems. Especially the communication with gateways, sensors and actuators of the different de facto standards can be a challenging part. Such a toolkit is provided by the openHAB UG [21]. OpenHAB is an open-source toolkit, which enables access to several home automation bus systems in order to monitor and control sensors and actuators, for example via Ethernet connections. In most cases such toolkits act as interface between the user and the underlying home automation bus system.

In addition, the ability to access the home automation system via TCP/UDP based protocols enables access to the system not only from the internal network, but also from external places via an internet connection, if communication ports are forwarded by the local internet gateway.

1.6. Requirements and Recommendations

From the investigation of disaster and crisis management, important elements as awareness, information representation and reaction capabilities can be identified, which are crucial for a comprehensive warning approach. Derived from these findings, functional and non-functional requirements can be defined for a disaster and crisis alerting approach employing home automation systems. The specification of requirements and technical considerations are defined in the following sections.

1.6.1. Functional Requirements

First of all to develop a comprehensive concept and prototype system, it will be necessary to consider the requirements of early warning as identified in section 1.3.4. , including user and technical aspects. The functional requirements are divided in the two sections, based on the system components and actors.

Disaster Information Service

In order to provide the smart homes with appropriate information on catastrophes and other incidents, the following functions are defined to cover the requirements for a reliable warning service.

- A platform or service accessible via internet is required, which acts as central disaster information source for the endpoints.
- Information about current incidents can be created, monitored, updated and archived on this platform. This service is owned and hosted by national authorities to ensure that information is trustworthy and valid.
- The provided information has to include valuable information about what has happened, which region is affected by the incident or what mitigation action are recommended.
- The platform requires a standardized communication interface to allow easy and lightweight access to incident information.
- The provided information has to be structured in a way that allows the clients easy interpretation and further processing.
- Related to endpoint owner’s privacy and anonymity, the provider of the disaster information service should have as little information as possible about smart homes consuming the service. This should protect owner’s privacy and prevent national authorities to collect sensible information.
- Therefore, it is recommended that the alarming service provides information only, so that the endpoints are required to ask for incident information and are urged to initiate the communication.

- Since communication takes place via an Internet-based connection, the data sent through this channel has to be protected. Therefore encryption of data to prevent alteration (e.g. man-in-the-middle attack) and digital certificates to proof the authenticity of the information provider have to be considered.

Endpoint/Middleware

On the smart home side, a middleware is required acting as interface between the disaster information service and the home automation system to ensure interoperability. This middleware has to implement the following features to perform as desired:

- The endpoint communicates with the control center system via an internet-based connection.
- It is necessary that the smart home knows its geographical location to evaluate if it is affected by the incident.
- It is possible to interpret risk mitigation actions based on the incident information.
- Based on the information of the incident, the endpoint can send instructions to the home automation bus system and monitor the state of sensors and actors.
- Successfully or failed instructions sent to the home automation system can be handled by the middleware.
- In addition to alert notifications, the owner of the endpoint should be notified by the endpoint that some risk mitigation actions can be performed. This requires also a kind of user confirmation to allow the execution.
- The owner of the system should be able to specify, which types of incident alerts can be received and which actions are allowed to be performed.
- The owner can view general information and current status of the endpoint via a simple graphical user interface.

1.6.2. Non-Functional Requirements

The following common non-functional requirements describe technical necessities to ensure that quality aspects are considered in the design of the communication infrastructure.

These requirements are listed in the following:

- Scalability
    - The disaster information service should guarantee acceptable performance, handling multiple requests even in large scale environments. Especially in assumption that this system is used by thousands smart homes.
- Reliability
    - Reliable communication between the control center and the endpoint has to be guaranteed (For example in case of transmission failure, time out, etc.)
    - Availability and redundancy of the disaster information service in case of failure. It should be avoided that the information service isn’t available or responds with delays because of overload.

- Interoperability
    - A standardized protocol or API for communication is required to ensure communication between various smart home systems and the disaster information service.
- Information Security
    - Confidentiality is primarily required for national authorities to access the disaster information service in a secure way. This can be achieved using authentication mechanisms (e.g. username and password) for access control.
    - Integrity – It can be ensured that information sent is not changed unintentionally. This requires cryptographic checksums in order to verify integrity of data.
    - Availability – The information is accessible when it is really needed. Also make sure to guarantee redundancy of information in case of transmission failures.

2. Distributed Information Systems – Design Aspects

The basis for a disaster alerting system employing home automation systems is a reliable and well-performing disaster information and communication infrastructure. In this section, approaches and good-practices are presented to address architectural requirements defined in the section earlier.

2.1. Communication Strategy

Before it is possible to think about the network protocols and other communication techniques, a basic application design decision has to be made, how the message transfer between the endpoints and the disaster alerting service take place. Basically, there are two ways in message transport, using a push or a pull communication specific model.

A push based approach indicates that the message transfer is triggered by the server [22]. If a specific event on the server occurs and conditions are met, the server send messages to its connected clients. For example, Radio and TV media. This assumes a permanent connection between the two members and the receiver has to be prepared to receive messages at the time of transmission. This could be problematic in case of disaster alerting if receivers are not online at the time of message transfer. So, an application logic has to be implemented to check if the receiver did receive the message and repeat the transmission in case of failure. A push approach can be highly scalable from the viewpoint of network-load. But such an approach produce a higher workload on the server-side, because of the need of session and error handling. From a security standpoint, it can be critical for clients when a system can push messages or even malicious commands from outside.

The pull model works the other way around. The server makes the information available via a common interface available. The server-side algorithms stores the information or messages, until the clients request this source. That is exactly the way how most web services works. This indicates that the clients are responsible for connecting to this interface and ask for new information. A pull mechanisms, also called *polling,* generate more transmission overhead, because the connection to the information source have to be established frequently [22]. This approach creates higher network and server load by default, if many clients send requests at the same time. One solution for this issue is to set a time interval for client requests by including an adjustable delay to distribute the requests in time. In general, the benefit of this pull-based approach is that the server-side don't have to worry about who is connected and if the client has got the messages. This behavior can also be seen as broadcast, because there is no restriction to requesting clients. This allows a lightweight and slim implementation of server-side application logic, supporting the aspect of vertical and horizontal scalability. Although it also means that the responsibility of connection and error handling has to take place at the client-side.

A pull-based communication approach seems to fit the needs of a disaster alerting system, because of the mentioned aspects. For an effective implementation, of such an approach, good-practices and style patterns exists which will be investigated in the following.

2.2. Communication Models

Client/Server Model

The client-server model is one of the standard concepts to provide and distribute data within a network. Basically, tasks are performed by applications which are divided into two main elements. Typically, client and servers communicate over a computer network on separate hardware. But it is also possible that both elements exist in the same hardware or system. A server can provide common services or information and share its resources with multiple clients. A traditional view, the client only requests resources from the server and does not share any of its resources with others. But depending on the application context, clients can also be used to provide server functionality as well. The communication initialization is done by the client which awaits incoming responses from the server. This principle is illustrated in the following figure:

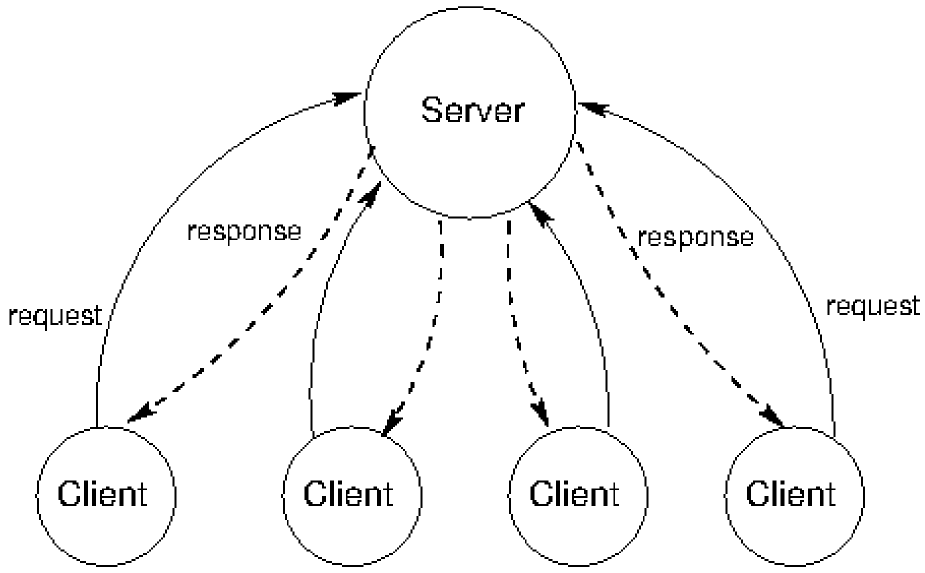


Figure 10: Client-Server Principle [23]

The concepts of client and server are powerful functional abstractions [23]. A server is simply a unit that provides a service, possibly to multiple clients simultaneously, and a client is a unit that consumes the service. The clients do not need to know the details of how the service is provided, or how the data they are receiving is stored or calculated, and the server does not need to know how the data is going to be used.

Peer-to-Peer Model

Another communication approach is introduced by peer-to-peer networks. Essential feature of peer-to-peer (P2P) infrastructure is decentralized networking. Decentralized means that no central control-, service- or data instance exists on the network, as it is the case using a centralized server. A peer is a network node acting as a client as well as server. A peer manages its own resources and allows access to it. Each peer is thus a client and a potential server and provides services and data for other peers. In contrast to the client-server principle, the communication is done directly between members of the peer-to-peer network. This behavior has some advantages due to flexibility and the possibilities of fault tolerance especially in view of scalability. Scalability and fault tolerance are central problems in single Client-Server architectures, which represent a basic motivation for realizing P2P networks. The decentralized approach can avoid single-point-of-failure and bottlenecks regarding communication paths and network load.

The decentralization concept in pure peer-to-peer networks can be described as follows [24, p. 28]:

- There is no central coordination. No central node exists controlling the interactions of peers with each other. Peer coordination is distributed.
- There is no central database, so no peer knows all resources of the system. These resources are distributed to the peers.
- No peer has a global view of the system. Each peer knows only its neighbors; only those peers where data exchange takes place.
- Peers and connections between peers do not have to be reliable.

The absence of managing servers within a pure peer-to-peer network implies that the members must organize themselves [24]. This self-organization is the biggest challenge of today's peer-to-peer systems. This includes the management of addressing structures and routing process. So, peer-to-peer networks generally implement some form of virtual overlay network on top of the physical network topology, where the nodes in the overlay form a subset of the nodes in the physical network. Data is still exchanged directly over the underlying TCP/UDP network, but at the application layer peers are able to communicate with each other directly via the logical overlay links. So, overlays are used for indexing and peer discovery, and make the P2P system independent from the physical network topology.

2.1. Scalability and Performance

Scalability is a performance indicator which describes the flexibility of an information technology infrastructure regarding varying load. As stated by Von Brauk and Neudert [25], some aspects have to be considered in building scalable client-server systems. The first principle deals with the scaling strategy itself. One possibility is to use a vertical (scale-up) approach. This is synonymous with adding extra memory or a more powerful processor in an existing server. The benefit of scaling up is that no additional servers have to be integrated, no software has to be installed and the IT architecture remain unaffected. This method is especially applied when no possibility exists to parallelize the application on several servers. The upgradability of a single node depends on the model or type of hardware, because of physical limitations. So using this technique the only remaining step is to replace the existing hardware with more powerful. This approach not ensure linear scalability, it also strongly depends on the parallelization of code. The alternative scaling strategy of horizontal scaling (scale out) is on a different level of granularity. Von Brauk and Neudert [25] note that any increase in load on an IT system, it should be possible to respond appropriate by adding new nodes, for example standard hardware or virtual machines. Apart from the cost aspect, this elasticity has the advantages that the power expansion can be implemented on the fly and without costly migration projects. Significant architectural challenges are generated, for example in terms of data consistency, availability and redundancy by parallel operation of identical servers and applications. Obviously, the realization of this kind of scaling has already to be considered at the application and infrastructure design. Another aspect of the horizontal scaling principle is to optimize the connection between resources and consumer.

Another problem in architectures is that data is tied to resources, such as session or connection data in a web server [25]. The session or connection can only be served by the processing instance. In the case of synchronous communication, in which an open transaction is waiting for the response of the writing service, resources such as threads, memory, and network connections are used. Bottom-up through the idempotent design of services and top-down by scheduling state-controlled communication at the borders of the system. Idempotent services deliver the same result even if the action is repeated. All read operations are idempotent. Idempotent services reduce the complexity associated with a messaging system, since they always deliver the same result even through multiple requests. Statelessness can also be promoted by terminating as many state-dependent communications as possible outside or at the borders of the system. This includes, for example, terminating TLS connections on the load balancer and implementing all requests within the system in an unencrypted manner.

Multi-Tier Architectures

Multi-Tier architectures allow a flexible setup of complex distributed systems based on the client-server principle. Basically, tier-architectures are built up in layers. The more complex the requirements the more layers are used. Such a modular design makes it very flexible to respond to future requirements and changes. In general, three types of layers are common in client-server architectures. These layers are described by [24, p. 26] as follows:

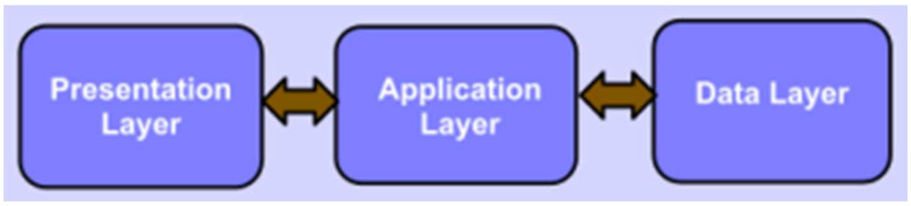


Figure 11: Multi-Tier architecture [26]

- **Presentation Layer**: The presentation layer is responsible for the presentation and retrieval of data that has to be processed. In distributed systems, there are two alternatives for the presentation of content. Thin-Clients are typically minimalistic versions with less powerful hardware and software equipment, only used for standard input and output tasks. Such clients only offer terminal functionalities to access services provided by the server. In contrast, fat clients are equipped with all the hardware and software components necessary for processing data on the client side. Fat clients, also called rich clients, implement the application-specific functionality, for example the user interface (presentation logic) and the logic of applications directly at the client.

- **Application layer**: The application layer provides the coordination of several professionally delineated parts of the business logic. The business tier is pulled out from the presentation tier to its own layer. It controls an application's functionality by performing detailed processing. The application layer makes logical decisions, evaluations and calculations and acts as interface between the two surrounding layers.

- **Data layer**: The data access layer encapsulates access to persistent data and the techniques used. For the persistent storage databases are often used, but this is not mandatory. It is also possible for small distributed applications to use the filesystem. When using databases for data exchange with the application layer, interfaces has to be implemented for the access to the database management system.

The use of these layers can result in 1-Tier, 2-Tier or Multi-Tier architectures. In 1-Tier architectures all of this layers are located in one system, for example this concept takes place in mainframe systems. In this structure, thin-clients are only used as input and output terminals. 2-Tier architectures are the typical client-server systems, where the presentation layer located at the client-computer. But it also can be necessary to implement parts of the application logic at the client side, whereas the basic application and data logic remains on the server side. In 3-Tier or Multi-Tier architectures an additional layer is used to separate the application logic and the data storage system. The application layer of this architecture is between the user interface and the database server and can be realized for example by transaction monitors, message services or an application server.

In building modern applications and distributed information systems, a multi-tier design is state of the art and best practice. The benefit about this concept is the possibility to scale one layer of the system without the need of changing other layers, for example application layer and data layer. A layered architecture supports the aspect of application scalability, but there are other methods to consider.
Depended of the architectural design of distributed systems, application functionality can vary at different levels, as shown in Figure 12: Client-Server functionality by design, related to [27, p. 334].

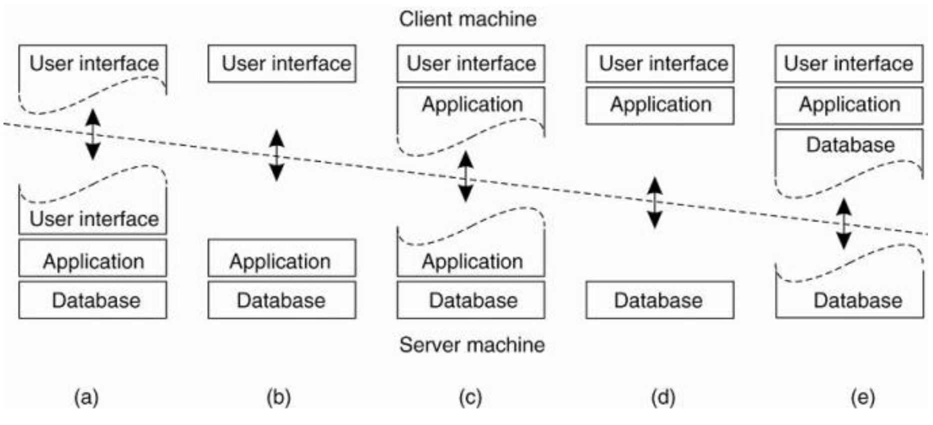


Figure 12: Client-Server functionality by design, related to [27, p. 334]

According to Mandl [27, p. 333], variants (d) and (e) are not very common in larger developments, but still can be found in many applications. They have the advantage that the client can access the database directly. In contrast, (a), (b) and (c) are todays most used variants developing distributed applications and systems. Besides this vertical distribution of application layers, a horizontal distribution can be realized, by placing the application and data layer on multiple server units.

In context with disaster alerting and information handling of the endpoint, a system architecture based on approach (d) will be recommended. The server provides information only, whereas the endpoint application logic is responsible to process the receive information about incidents.

2.2. Incident Information

As identified in the requirements, a standard information format have to be provided that specifies, which information have to be included in alert messages. OASIS [8] established a common notation, which has been specially developed for the information exchange of disaster information. The CAP format is compatible with emerging techniques, such as Web services, and include the following capabilities [8]:

- Flexible geographic targeting using latitude/longitude shapes and other geospatial representations in three dimensions
- Multilingual and multi-audience messaging
- Phased and delayed effective times and expirations
- Enhanced message update and cancellation features
- Template support for framing effective warning messages
- Facility for digital images and audio

The message is presented and structured in a format, which is ideally suited for machine-to-machine communication. The following figure illustrate the structure of a CAP message:

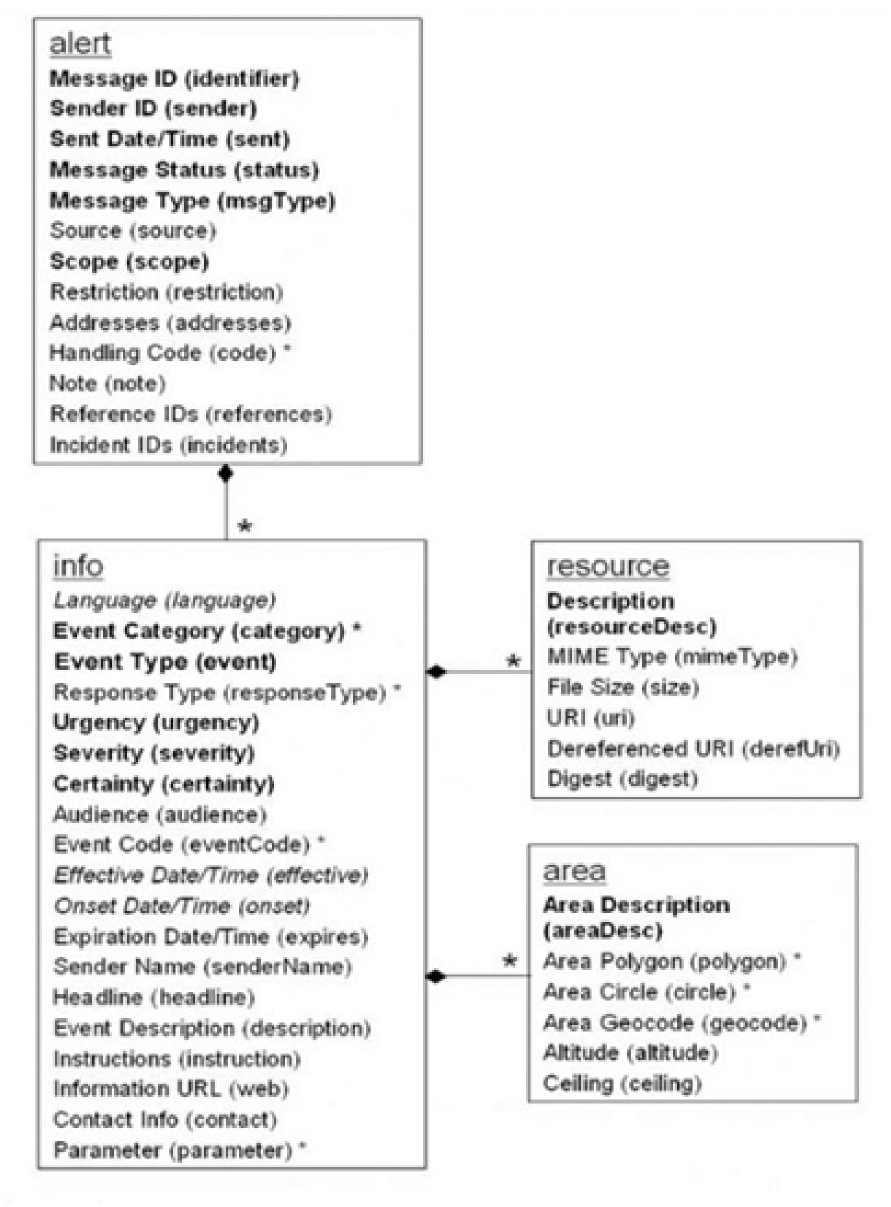


Figure 13: CAP message structure, related to [8]

A message consists of four basic elements. In the *alert* segment, all information about the purpose, status of the message itself is contained. The *info* segment provides space for information about the event, for example type of the incident, urgency or instructions. In the r*esource* segment, it is possible to link additional textual or audio information related to the info segment of the message. All information about the occurrence, location and range of the event can be described in the *area* segment. As shown in Figure 14, all bold representations are mandatory for a CAP specific message. Elements in italics are filled with default values if the element is not present. More details of the Common Alerting Protocol Version 1.2 can be found on the OASIS webpage. This standard is an excellent basis for the alert message format for the disaster alerting approach employing home automation systems.

2.3. Communication and Information Exchange

This section deals with communication methods and mechanisms for information distribution. Important aspects and design decisions are highlighted to support a secure and reliable network communication between the home automation endpoints and the source of disaster information.

2.3.1. Communication Protocols

The TCP/IP reference model, which is an abstraction of the OSI reference model, was developed by the Internet community. It covers layer one to four of the OSI reference model. This model is seen as the de-facto standard of internet-based communication.

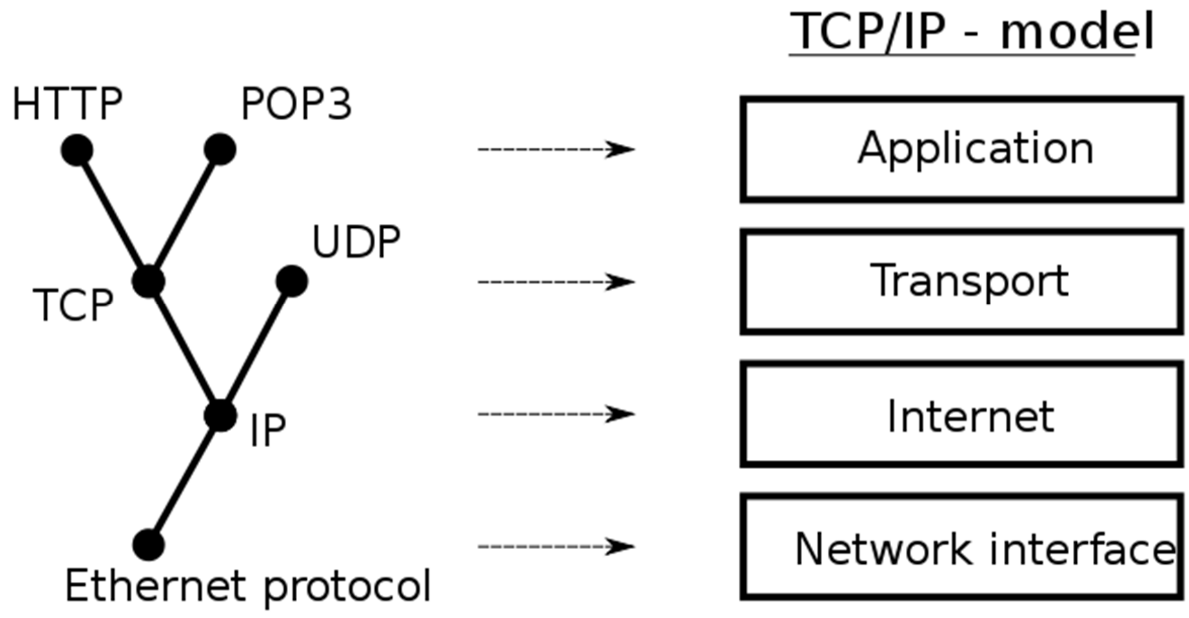


Figure 14: TCP/IP reference model [28]

In contrast to the OSI reference model, the TCP/IP reference model consists of four layers, as seen in Figure 14. The TCP/IP reference model is designed to enable data exchange beyond the boundaries of local networks also called “Internetworking" [28]. The application layer includes all protocols that work with application programs and use the network infrastructure for the exchange of application-specific data, for example HTTP, FTP, SMTP or DNS. The transport layer establishes an end-to-end connection. The main protocol of this layer is the Transmission Control Protocol (TCP). It creates connections between two network devices for secure transmission (TCP 3-Way-Handshaking) of data streams. The User Datagram Protocol (UDP) is a connectionless transport protocol without any secure data transmission mechanisms as provided by TCP. Due to this fact data packets can be exchanged relatively fast between two hosts. Therefore, it is used where quick delivery is more important than the reliability in relation that the data has arrived correctly and completely.

The knowledge of the main properties and behavior of the two protocols can be essential for designing distributed communication infrastructure and applications. Especially in large scale environments with a high traffic rate.

The major difference between the protocols are listed in the table below:

Property	**TCP**	**UDP**
Connection	Connection-oriented	Connectionless
Usage	Suited for applications that require high reliability, and transmission time is relatively less critical.	Suitable for applications that need fast, efficient transmission.
Ordering of data packets	TCP rearranges data packets in the order specified.	UDP has no inherent order as all packets are independent of each other
Speed of transfer	Slower than UDP	Faster, no error-checking for packets.
Reliability	Guarantee that the data transferred remains intact and arrives in the same order in which it was sent	No guarantee that the messages or packets sent would reach at all
Header Size	TCP header size is 20 bytes	UDP Header size is 8 bytes
Weight	Heavy-weight. TCP requires three packets to set up a socket connection, before any user data can be sent.	Lightweight. There is no ordering of messages, no tracking connections
Data Flow Control	TCP does Flow Control. Handles reliability and congestion control	No option for flow control
Error Checking	TCP does error checking	UDP does error checking, but no recovery options
Acknowledgement	Acknowledgement segments	No Acknowledgment
Handshake	SYN, SYN-ACK, ACK	No handshake (connectionless protocol)

Table 3: Comparison TCP vs. UDP Protocol, related to [28]

As seen in the comparison matrix above, UDP is quite faster than TCP connections, due to the fact that no transmission control is given by design. Therefore, control and security mechanisms has to be implemented on the application layer. UDP's stateless nature can be useful for servers that answer small queries from huge numbers of clients, but do not rely on a secure connection. The transmission control protocol focus on secure and reliable communication by implementing security mechanisms like handshaking. This should guarantee that information is transmitted successfully. But on the other hand, the secure transmission of TCP generates a higher network load as in case of user datagram protocol. Another positive effect of the TCP protocol is that data can be encrypted using Transport Layer Security (TLS).

In conclusion, UDP protocol can be theoretically used for data transmission, but only under the condition that a CAP record fits in one UDP datagram. Otherwise it would be necessary to implement a kind of sequence order mechanism to ensure that data arrives correctly using UDP. This sequence order problem of the UDP protocol is among other things a fundamental reason why TCP is better suited, because of security capabilities and trustworthy transmission of information between two communication partners.

2.3.2. Reliable Data Transmission

Depending on the domain of the application, it can be necessary to encrypt data in transmission to avoid man-in-the-middle attacks. A common practice in internet communication is to encrypt data via Transport Layer Security [29]. The latest version is TLS 1.3 which was introduced in 2018. TLS is an encryption protocol typically used for secure data transmission, for example over Internet connections.
The principle how it works can be described as follows: The client connects to the server. The server authenticates itself against the client with a certificate. The client checks the trustworthiness of the X.509 certificate and whether the server name matches the certificate. Optionally, the client can authenticate itself against the server with its own certificate. Then either the client sends the server a secret random number encrypted with the public key of the server, or the two parties compute a secret with the Diffie-Hellman key exchange. A cryptographic key is then derived from the secret. This key is subsequently used to encrypt all messages of the connection with a symmetric encryption method and to ensure message integrity and authenticity through a message authentication code. The advantage of the TLS protocol is the ability to implement any higher protocol based on the TLS protocol. This ensures independence of applications and systems [29].

The question that arises in context with a public disaster alerting system is, if it is required to encrypt data in transit at all. The basic goal is to establish a service able to alarm and warn people about upcoming threats. This implies that incident information have to be public because of its domain. So, incident information transmitted via internet actually do not need encryption, because it do not contains no sensitive information which have to be protected. It is more the contrary case. But it have to be ensured that this data have not been altered on the way to its addressees. Integrity and origin of the data are more in focus. Hence, checksums and digital signatures play a more vital role in context with a disaster alerting system.

2.1. Domain Name System for Information Exchange

As is generally known, a Domain Name System (DNS) is a name service, an online distributed database system, for example converts domain names into IP addresses. DNS is based on the UDP protocol (port 53) and is essential part of internet communication [30]. The advantage about DNS is decentralization aspect and the way how the information is provided to requesting clients.

Basically, name resolution can be achieved by recursive and iterative DNS queries, illustrated in the next figure:

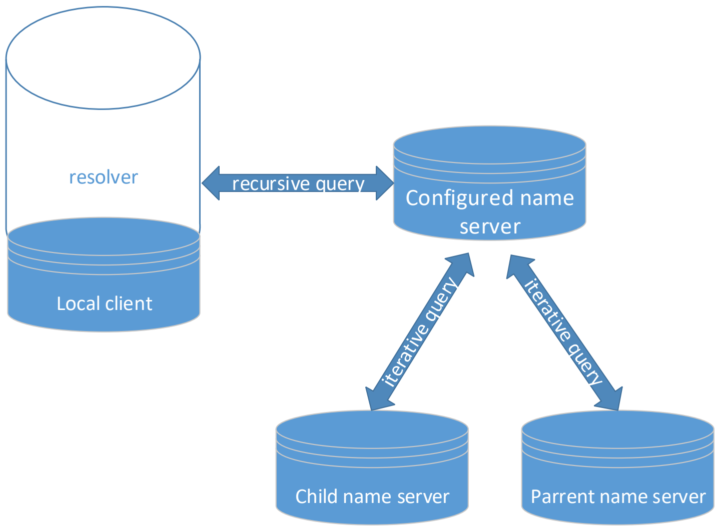


Figure 15: Name resolution using DNS, related to [31]

With a recursive query, the DNS server respond to the client with either the requested resource record or an error message that the domain name or record does not exist. If it is the case that the DNS server does not have the information requested by the client, it queries other server until it gets the information or until the name query fails. So, recursive name queries are generally made by DNS clients to a DNS server. An iterative query allows the DNS server to return a referral, which is a pointer to a DNS server authoritative for a lower level of the domain namespace. The DNS client can then query the DNS server for which it obtained a referral. It continues this process until it locates a DNS server that is authoritative for the queried name, or until an error or time-out condition is met. This kind of query is typically initiated by a DNS server that attempts to resolve a recursive name query for a DNS client [32].

In addition, DNS caching is another important mechanism for efficient name resolution. DNS caching retains the result of a recursive query for a specific time in the DNS server's local cache before responding to identical DNS queries from other clients without having to re-request in the name servers. The length of time that a cache entry is valid is determined by its TTL (time to live) value. The TTL value is specified by the authoritative name server of the corresponding zone. Generally, caches are implemented for example in the resolver of the operating system, the name servers of the Internet access provider, and some applications such as web browsers. The of cached data at different locations is to relieve the authoritative name server and saving time, because a request can respond faster from the cache than by re-requesting the responsible name server [32].

The existing and already usable communication infrastructure of DNS (including caching) is an interesting option for providing data to clients in this context. Therefore, a conceptual model was built to illustrate such a possible approach. As shown in
Figure 16, information about the occurrence of incidents could be provided by using Resource Records (RR), for example with TXT records which contains the information needed. Smart homes could use DNS queries to request cached information about current incidents. The benefit is that information is decentralized and principally already available on ISP's DNS server cache, if a request was already made from a client using the same DNS server. That means, only one DNS query has to be made to get the information from the server of origin within ISP's DNS environment. This information will be stored in DNS cache and shared with any other clients requesting (consider the TXT's TTL value).

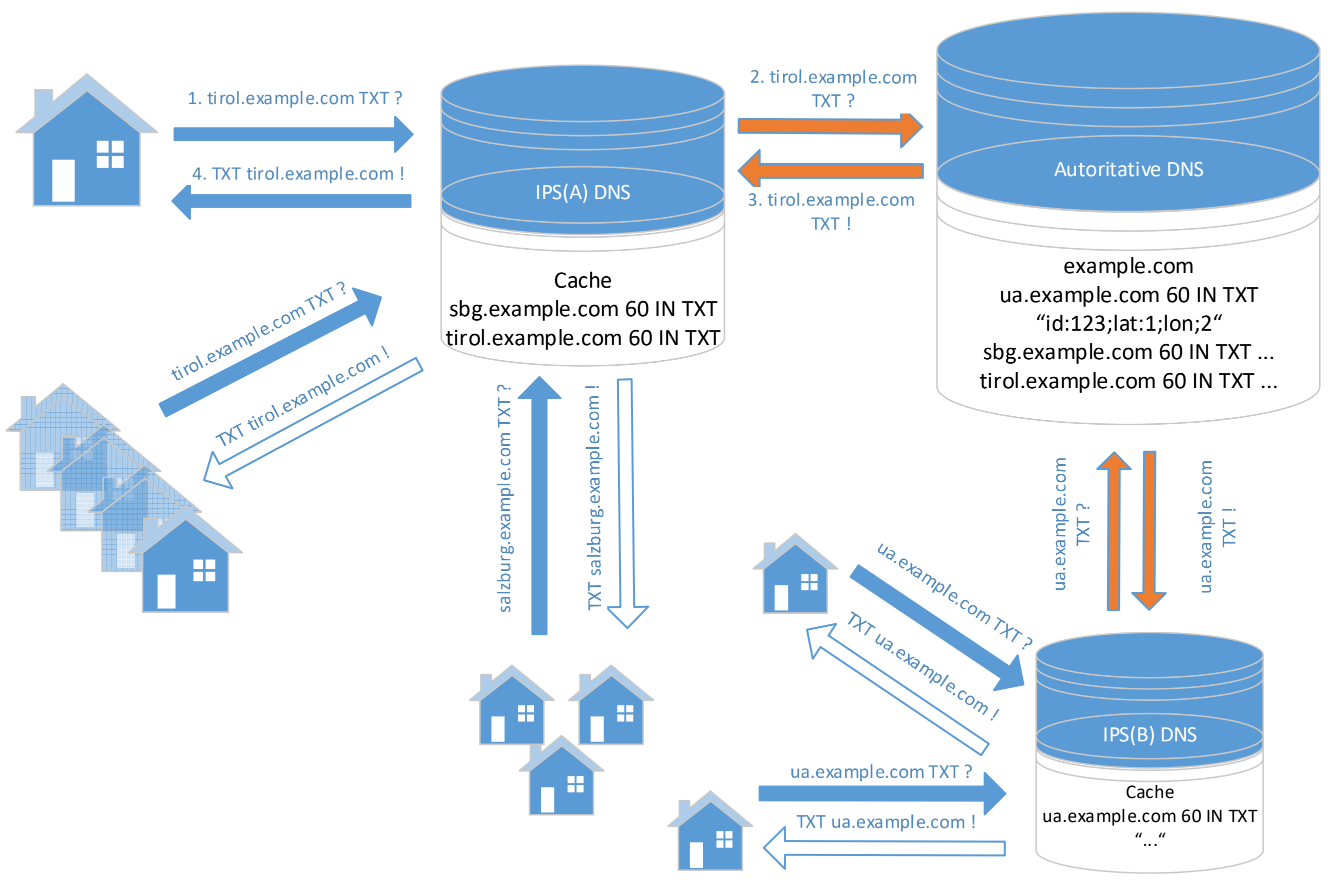

1. tirol.example.com TXT ?
4. TXT tirol.example.com !
IPS(A) DNS
Cache
sbg.example.com 60 IN TXT
tirol.example.com 60 IN TXT
2. tirol.example.com TXT ?
3. tirol.example.com TXT !
Autoritative DNS
example.com
ua.example.com 60 IN TXT "id:123;lat:1;lon;2"
sbg.example.com 60 IN TXT ...
tirol.example.com 60 IN TXT ...
tirol.example.com TXT ?
TXT tirol.example.com !
salzburg.example.com TXT ?
TXT salzburg.example.com !
ua.example.com TXT ?
ua.example.com TXT !
ua.example.com TXT ?
TXT ua.example.com !
IPS(B) DNS
Cache
ua.example.com 60 IN TXT "..."
ua.example.com TXT ?
TXT ua.example.com !

Figure 16: Information exchange using DNS

So, using existing DNS infrastructure seems to be a possible approach to distribute incident information to smart homes, because information can be decentralized. But there are some restrictions to consider. Such an approach is only suitable when a small amount of data will be transmitted, because of the size limits of the resource records. Moreover, time critical content would be probably problematic for dynamic and frequently updated data as it is required in disaster alerting. Actually, frequent record changes are not a really a technical issue, because the update interval can be controlled via TTL values. The problem that can arise is more of an organizational nature. ISPs are not enforced to accept very small TTL values. Smaller TTL values increase the update frequency on DNS servers. In some cases, they are ignoring TTL values of records and accepting only values higher than 3600 seconds, in order to prevent a misuse and potential overload of the system. In addition, DNS is working over UDP. As is generally known, UDP is a stateless protocol. So, there is no guarantee that messages or packets sent would arrive in the correct order. Moreover, the authenticity and integrity cannot be guaranteed unless Domain Name System Security Extension (DNSSec) is used, for example signing the content. At the one hand, DNSSec provides protection mechanism to verify if the contend has changed during transmission and reduce the risk of DNS cache poisoning.

Using DNS infrastructure would probably work to provide meta-data of incident. No additional infrastructure and protocols have to be implemented to provide data to the endpoints. But due to the mentioned limitations, some compromises have to be made, which could restrict the efficiency and reliability of this approach.

2.1. Anycast Communication

Anycast is a communication and routing mechanism where a single destination address can have several endpoint addresses. While unicast addresses a unique target, broadcast addresses all systems in the subnet, and multicast addresses certain groups of nodes, anycast communication always finds the closest possible target via routing information [33].

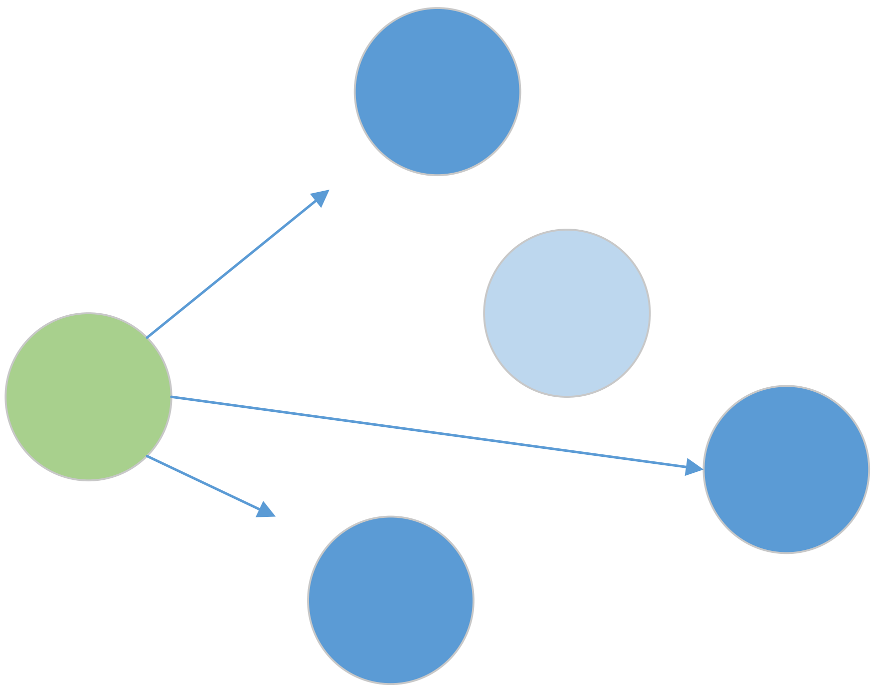


Figure 17: Anycast Communication Schema, related to [33]

In an abstract view, endpoints receive the same IP address and propagates a corresponding route via a routing protocol, for example as implemented by the BGP (Border Gateway Protocol) to route internet traffic. In the event of a failure or inaccessibility, the route disappears and all subsequent packets are routed to another server. The desired service can thus also be provided if one or more servers fail. This increases the availability. In contrast to multicast environments that generate packets for all group members, the sender creates only one single packet. A packet with an anycast address is only sent to its nearest interface, depending on the entry in the routing table of the router closest to the sender. Important advantages of anycast are shortened access times through shorter routing routes, load balancing by using different systems depending on the location of the sender and scalability.

Basically, the general concept of anycast would be interesting for distribution data. But the realization of anycast in an inter-domain scenario requires its own routing infrastructure, as it would be the case in context with this disaster alerting approach. An implementation would require lots of routing equipment and configuration tasks to integrate smart homes is such a scenario. The implementation of a native anycast approach would result in high configuration effort and hardware requirements in order to create a working communication infrastructure. Therefore anycast based approach is not really applicable for end-to-end communication via internet. Nevertheless, anycast can be indirect part of a possible disaster alerting approach, because it is increasingly used in Content Delivery Networks (CDN).

2.2. Content Delivery Networks

In a scenario, where all resources are bundled geographically in one place, the network topology can become the next bottleneck. A possible solution for this restriction is given by content delivery networks. Besides the horizontal scaling in content delivery networks, nodes are also distributed geographically in order to optimize latency and throughput by serving content from the closest geographic location.

Horizontally and geographically distributed nodes can be seen as simple caches of a central server or instance. At this stage Von Brauk and Neudert [25] advices that this distributed caching technique is effective for architectures, where read-only access takes place and no client-write action is needed, because the necessity of a write feature can easily lead to consistency problems. Regarding to a central disaster information and alerting service, there is no need that clients perform any write requests, because the service is designed provides information only.

Key features of content delivery networks, according to Bhatia [34]:

- The latency of delivering content is reduced giving very fast throughputs.
- Network load is optimized by caching near to the point of serving content and not consuming long distance bandwidth.
- Content is delivered with optimum performance with minimized dependency on middle networks.
- Ensures that there is no single point of distribution, hence traffic serving can be optimized by multiple sources.

Always in mind that in a future scenario thousands of clients may access this service, utilization of Content Delivery Networks is a modern approach to provide information to the clients in a scalable, redundant and resilient way. Especially the combination of load balancing and geographically distribution of information hosting nodes is necessary requirement for the disaster and alerting service.

The architecture described above can be extended to multiple sites for more efficient information distribution and traffic optimization. In a nationwide scenario, where thousands of smart homes exists and are possible candidates, a simple HTTP request/response approach could lead to a high traffic rate at the server-side. For example, in Austria 3.865.000 private households exists in 2016 [35, p. 20]. Theoretically, all these are potential candidates using this service and requesting incident information. Even with an optimized request distribution mechanism, this would still lead to 64.433 requests per seconds. The more requests arrives on one site, the higher is the probability that the response time will increase to several seconds or lead to service outages because performance issues.

The figure below is an illustration of one possible solution implemented with Content Delivery Network nodes to spread the load to various nodes and sites.

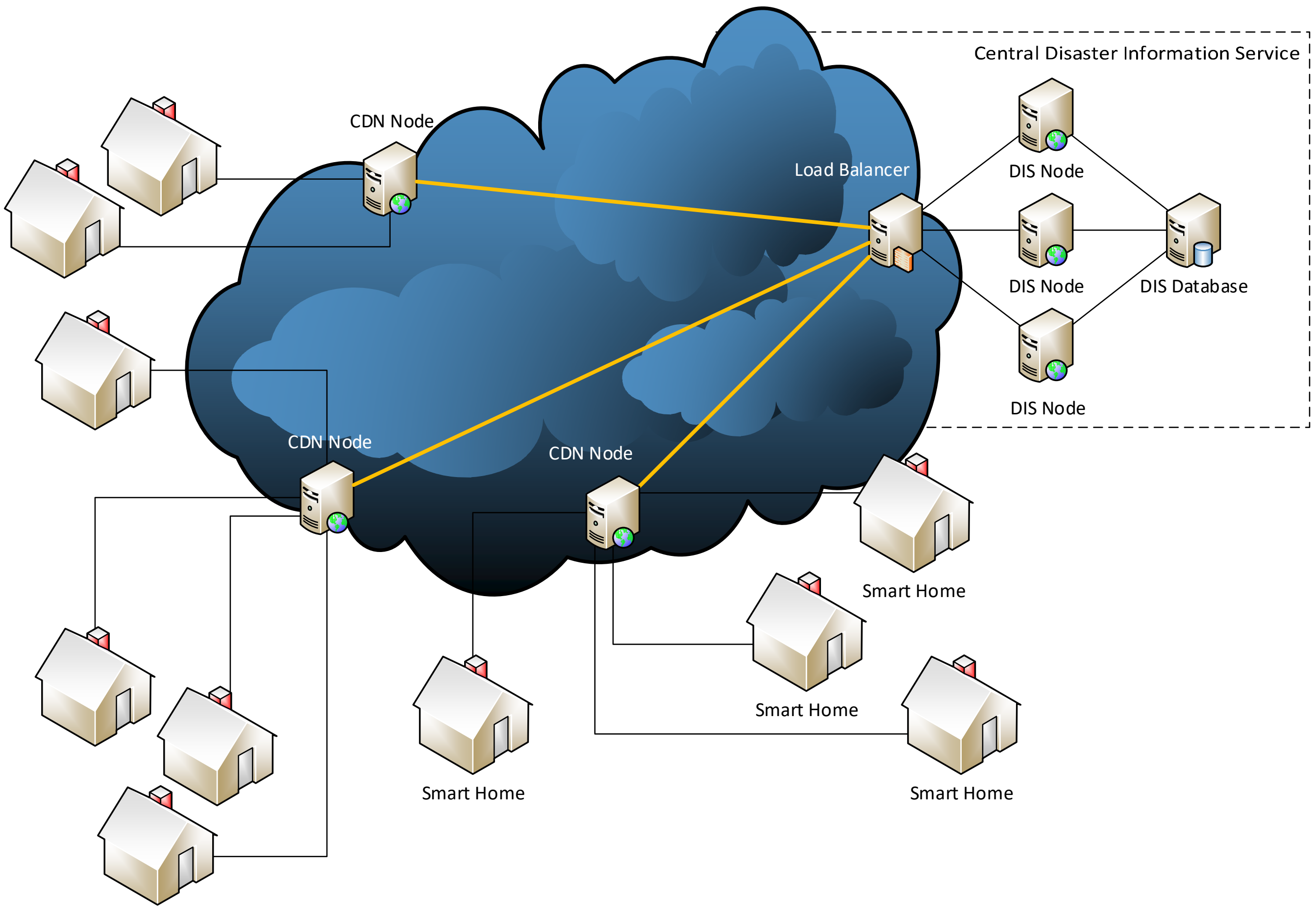


Figure 18: Information distribution with a Content Delivery Nodes

The underlying technique used by most CDN providers is based on a combination of DNS resolution and anycast-routing mechanisms [34]. So, if a user's request reaches the CDN, there are different ways how the request will be treated:

- If the requested files are not yet in the cache of the CDN PoPs (Point of Presence), the CDN loads it from your server to the PoP and then back to the requesting client.
- If it is the case that the requested files are (based on rules) in the cache of the PoP, the response is sent directly from there. No further request to the root server is required. When the time to live (TTL) of the resource has expired, the CDN systematically requests a new copy from the origin server.
- If the requested source is either dynamic or not in the cache rules, the CDN directs the request to the origin server, which sends the requested data directly to the customer.

Nevertheless, centralized pull-based communication approaches requires clients to frequently ask the source of information. This can lead to high network traffic, even information at server-side hasn't changed. Because of this fact, other methods should be considered, in order to provide propagate information changes in a more decentralized way.

2.3. Peer-to-Peer Communication Aspects

An alternative or even an extension to centralized client-server approach can be achieved by peer-to-peer communication. In context with the disaster information system peer-to-peer data exchange could be useful to share incident information with others and lower the network load on the centralized instance. Peer-to-peer information sharing capabilities can be illustrated by taking a look at the BitTorrent [36] concept.

Some basic characteristics of BitTorrent protocol used for file sharing:

- Files can be downloaded from multiple sources (called seeders) at the same time.
- Files are identified by a unique hash, which is used to find new sources of a file.
- The integrity of files is verified by checksums
- Downloaded files are shared by the client after being scanned to speed up file propagation.

BitTorrent is one, of the most popular peer-to-peer protocols, which is used for file sharing. To be able to distribute files with others, small .torrent files are created, which contain meta-information of the file to be shared. In addition to mandatory information such as `name` and `size` of the files, size and a list of checksums of segments of the data to be downloaded and additionally tracker contact details can be stored [37]. Torrent meta-info (of the file to be shared) consists of several attributes. A list of trackers that are used to store contact information (IP-Addresses and ports) of peers downloading or seeding file content. The number of file segments and length of these objects. It also includes the hash value of the file segment to verify integrity of the downloaded content. Based on these information a unique 160-bit *infohash* is generated.

An illustrative example of a torrent file:

```
{
	name : '<file to share>',
	pieces:[
		'111111111111111111111111',
		'222222222222222222222222',
		'333333333333333333333333'
	],
	announce: 'http://tracker.example.com'
}

Info Hash: 89798aeaf98ae8fa98ef8a7f98ae22134
```

Listing 1: Torrent meta-information

Because static torrent files are not very handy to exchange in order to enable other peers to join the swarm and download the same file, magnet links have been developed. A magnet link can only contain the *infohash* of a torrent and an optional list of trackers which can be contacted.

Advantages of utilizing a peer-to-peer information sharing approach in contrast to centralized systems:

- Fast downloads: Every downloader (peer) also became an uploader (seeder)
- Decentralization: No central point of failure

The benefits of peer-to-peer information sharing are obvious, but also challenges have to be accepted, in order to build a peer-to-peer communication system. Challenges in peer-to-peer communication:

- Peer Discovery: Every endpoint has to know where it can download the desired information
- Peer Connectivity: A communication channel have to be established to other endpoints, in order to download and share information.

2.3.1. Peer Discovery and Distributed Hash Tables

To discover and exchange peer connection details with others, typically trackers or a Distributed Hash Tables (DHT) are used today, for example implemented in the BitTorrent protocol. Trackers are centralized public servers, which are contacted by peers to make announcements in order to gather contact information of other peers. They manage information about one or more files through torrents and a client who learns from the tracker who else is downloading and distributing the file. As soon as a client has received a small piece of the file and has verified the checksum, it reports this to the tracker and can already pass on this file piece to other clients [36].

With the BitTorrent Extension Protocol 5 (BEP 5), Distributed Hash Table are introduced which are based on the Kademlia algorithm to utilize tracker-less peer discovery [38]. Using Distributed Hash Tables is more a decentralized approach aiming to store contact details on several participating nodes without using centralized trackers. In other word, each node as specified by BEP 5 can act as a tracker for its known nodes in a decentralized manner. Using DHT in combination with trackers (which can be single point of failure) are common practice today. Usually, discovery approaches operating over UDP to exchange peer contact lists. Whereas the transmission of data requires a more reliable protocol by using TCP.

According to BitTorrent's specification, a *peer* is a torrent client implementing the BitTorrent protocol and operating over TCP for data exchange between peers. A *node* is a client/server entity controlled by the peer and connected to the DHT overlay network via UDP. Hence, the node is actually used for peer discovery. So, a node can be identified by an ID. The node ID is a random number from the 160-bit space as BitTorrent infohash. In order to compare two node IDs and an infohash for closeness an algorithm for measuring the distance will be used. Participating nodes have to maintain contact details for a small set of other nodes in a routing table. The accuracy of the routing table gets better as the distance between the node IDs of others gets closer to the node ID of its own.

The distance calculation, as used by Kademlia DHT [38], is based on XOR operations:

$$distance(A,B) = ||A \oplus B||$$

Formula 1: Kademlia DHT XOR operation [39]

The result is an unsigned integer, indicating the smaller the value is the closer is the node. So, if a node search other peers for a given torrent infohash, it compares the infohash with the node IDs contained in the routing table. In a first step it contacts known nodes with IDs closest to the infohash and make a request for the contact information of peers downloading the torrent. If a contacted node identifies peers in its own routing table, it answers with the contact details of these peers. Otherwise the node responds with contact information of other nodes (maintained in its routing table) which are closer to the torrent infohash. The querying node tries now to find other nodes until it cannot find any other nodes closer to the target infohash. In case it cannot find any node that are closer to the already known it inserts the peer contact information for itself [38].

The next figure illustrates the routing concept based on Distributed Hash Tables:

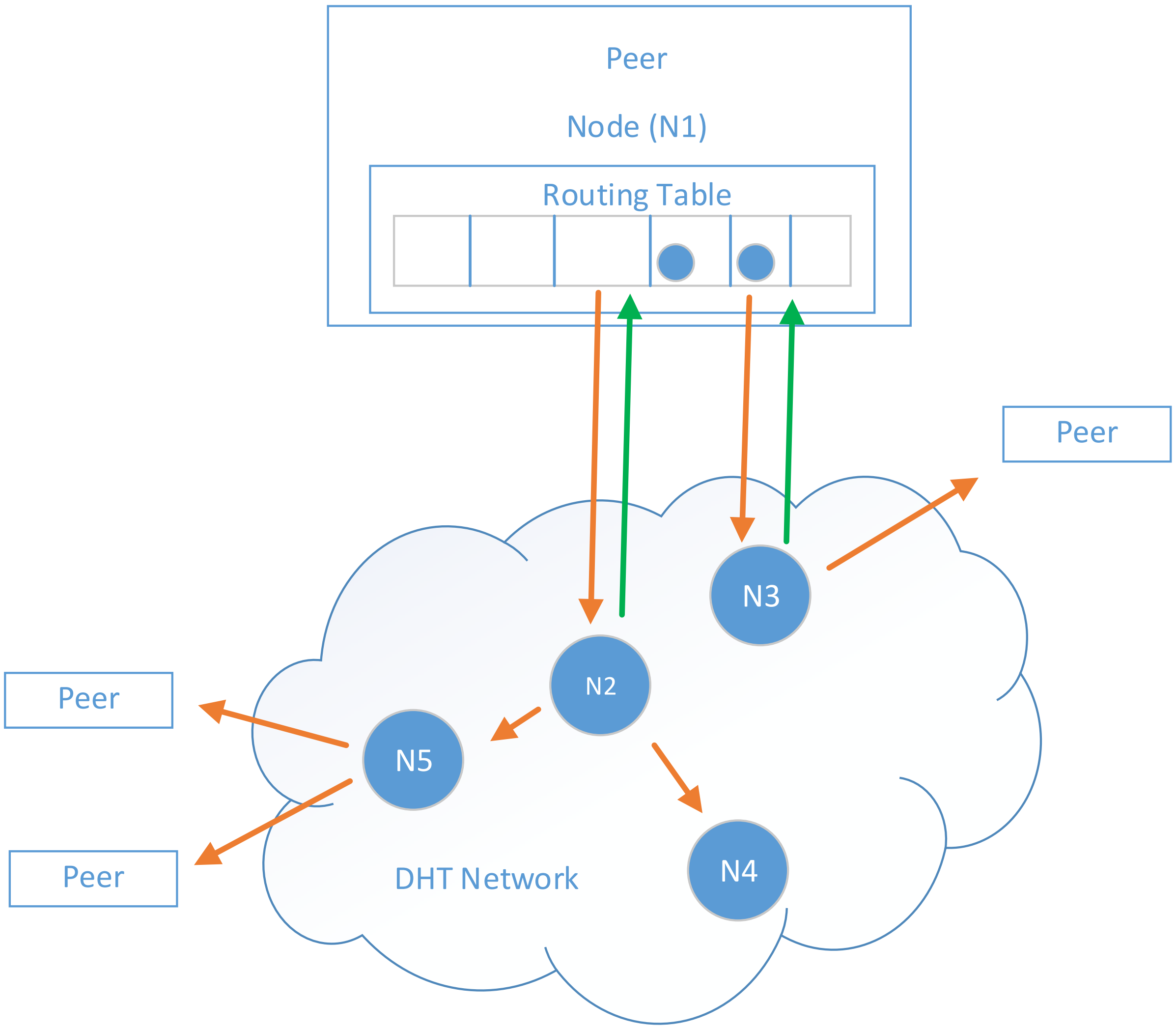


Figure 19: DHT discovery and routing concept, related to [40]

The goal of DHT is to maintain only nodes which are online and able to respond to queries from other nodes. So, if a node is not able to respond within the last 15 minutes its status is set to unknown. Each unknown node in the routing table is pinged in order to verify if it is online. If pinged nodes failed twice to respond the node will be discarded and replaced with other good nodes. Nodes are stored in so called buckets which are covering the node ID space from 0 to 2^160 and a bucket can hold K nodes (in BitTorrent 8 nodes) [38]. If a bucket is filled with known good nodes, no more nodes are added. But if the bucket contains the own node ID it is spilt into two new bucket. The nodes of the old buckets are distributed among the two new nodes. Buckets with inactivity of 15 minutes (no ping responses, node added or replaced) needs to be checked to verify that info of the nodes is the latest.

2.3.2. Connectivity

As a peer has received information about all seeders of a given file, it can start to connect to the remote peers. But first it have to be ensured that the seeding peer is reachable. In case of peers are used to communicate over internet, additional enabler has to be considered. Usually, BitTorrent protocol operating on default port 6881-6889. Because most peers are located behind a firewall or a NAT router because of limited address space of IPv4, some mechanisms are required to make each other peer visible and reachable. One approach is to use the UPnP (Universal Plug and Play) and NAT traversal techniques, described by Hu [41]. This mechanisms enables device in a local network to configure port forwarding on UPnP capable routers and gateways automatically or establish connections using hole punching techniques depended on NAT configuration. Otherwise to ports has to be configured manually in order to allow incoming connections to a peer behind a firewall/NAT.

2.3.3. Information Changes

Another challenge arises in context with a reliable disaster alerting system:

- Information changes: It has to be ensured that the shared information is the latest version published by an authority. Therefore, participants in a peer-to-peer network have to be notified to download and seed the latest version.

As it is given by design, only a trustworthy instance should be able to publish incident information. And this information is constantly updated. So this behavior could lead to versioning problems for content distribution in peer-to-peer based approach. Changes of content require changes to meta-information (new .torrent file), which has to be propagated to interested clients. With BitTorrent Extension Protocol 9 (BEP 9), an approach is introduced in order to download a file via BitTorrent and find related peers without the need of downloading a related torrent file first [42]. A so called magnet link is a URI standard for hyperlinks that point to files. It contains at least a unique identifier (infohash) of the torrent file. Therefore, magnet links are predestined to extending discovery features using distributed hash tables. Nevertheless, these information has to be published and distributed to peers first. Possible updates to incident information require peers to frequently look for changes, e.g. on a predefined time interval. This can be tricky especially when information which should be shared changes frequently.

One approach in order to provide new meta-information (e.g. torrent-files or magnet links) for a file/topic of interest is to use additional services, e.g. renew and publish magnet link (containing the new infohash value) via web-sites. But this requires in turn centralized service for initial meta-info exchange again.

Notification via DHT networks

However, also more decentralized approach was introduced by the BitTorrent team to address this challenge. A draft of a new BitTorrent Extension Protocol (BEP 46) has been release, which allows to updating torrent met-information using mutable content based on DHT [43]. This extension enables updating torrents based on data stored in the BitTorrent DHT instead using for example RSS feeds

(Really Simple Syndication) or web services. The initial seeder can notify others when the torrent is updated and point to the new information, via mutable DHT items. The goal of this solution is to allow publishers to serve content via a single torrent (magnet link), where it's content might change over time. Peers which are interested in the publisher's changing content can retrieve these updates by simple get request using the same torrent meta-info.

As described by Norberg [44], the goal of a decentralized RSS feed like service is to create a collection of torrents to avoid single point of failure. The DHT extension featuring *put* and *get* functionality in order to store and receive random data (with a maximum size of 1000 bytes) in the DHT network. Key-Value pairs can be stored either as immutable item (SHA1 hash of the torrent), or mutable item. Immutable items under a published infohash cannot be modified. Whereas published mutable items can be modified over time, without the need of changing the published contact point associated with the mutable item. In order to achieve that only the original publisher can change items, values are signed using publisher's private key. Mutable items are published under the SHA-1 hash of the public key [45].

So, peers requesting for publishers content only need to know the publishers public key to retrieve mutable payloads, verify integrity and version from the DHT network.

Publish and Consume Content

The BitTorrent Extension Protocol 46 described by Matteis [43], allows publisher nodes in the DHT to notify other nodes about updates on torrents. The payload v holds a key-value pair with key *ih* (infohash) and the value of the infohash. These payload can be consumed by other nodes with the *targetID* (SHA-1 of public key). To detect changes of property *v* constantly requests have to be made. If changes to *v* are found, the torrent can be updated based on the infohash from the response. Both publisher and consumer are urged to publish mutable items via put requests to keep them alive in the DHT network. In addition, as recommended by Norberg and Siloti [45], nodes with interest in the topic should re-announce the downloaded payload, because items can expire in the DHT network. This could be achieved by make *put* request of the item received. This would keep items in the DHT even the original publisher disappears.

This approach, publishing and consuming small amount of data via DHT network can be used to extend information distribution capabilities in a decentralized way. Further details of an integration of peer-to-peer capabilities using the millions of node based BitTorrent DHT network will be described in section 3.6.

2.4. Highlights and Comparison

This comparison of centralized and decentralized approaches highlights differences in behavior and functionality in context with the non-functional requirements of the envisioned disaster alerting system.

Reliability and Scalability: Reliability and Scalability (in sense of redundancy) and availability can be optimally supported due to decentralization, since each peer can practically operate as an information source. This is a big advantage, especially when gigabytes of static data have to be shared with others; several sources (peers) are available to seed pieces of already downloaded content, instead of a typical single source client/server approach. But, several techniques, like multitier architecture, load balancing and distributed service nodes can be implemented to create horizontal and vertical scalable centralized information systems.

Information Changes: Another challenge in peer to peer networks is the need to ensure that only the latest version of the data is available to all its clients. Whereas dynamic content can be problematic, for example when information about current incidents constantly changes, which makes frequently updates necessary. This can create a content versioning problem in the peer-to-peer network. Client-Server approach using a web-service can better address these challenge, because only one contact point is visible and available requesting clients, but also techniques are available to build a solution in terms of peer-to-peer network.

Connectivity: In assumption that peers are typically behind a firewall or NAT router, network configuration changes (opening incoming firewall ports on the home network) have to be made at endpoints site, in order to enable the provision of the information of a peer since the connection initialization takes place externally. Although, there are some NAT traversal mechanisms (e.g. TCP/UDP hole punching) to enable establishing bidirectional connections between peers in different networks. But this requires additional (central) services for signaling and even to relay data in case direct peer-to-peer connections do not work. A centralized approach does not really facing such a problem, because in a view point of an endpoint outgoing connections are not blocked in the most cases. So no configuration changes are required.

Security: In sense of security, the management of many different information sources is more problematic to ensure confidentiality and integrity of the information, as it will be the case in a centralized client/server approach. Each peer creates an attack surface, which makes it easier for intruders to take control of these information source or deliver malicious content to others within the peer-to-peer network. Of course, mechanisms are in place, for example integrity checks (hashing), digital signatures to verify the publisher or encryption techniques, but in turn this will create additional implementation effort. Security aspects in client/server architecture are easier to realize and manage compared to peer-to-peer networks, because of the centralization aspect.

While peer-to-peer networks provide advantages over centralized client-server systems in terms of scalability and availability, client-server systems can better address challenges such as connectivity, integrity and versioning, because of the centralization aspect.

Based on the requirements and challenges for a disaster alerting system, some kind of centralization is inevitable to create a reliable system. Because smart homes are urged to frequently request for new incident information even in peer-to-peer communication. The next section will introduce a concept of a disaster alerting system leveraging client/server and peer-to-peer communication techniques.

3. Conceptual Design of the Disaster Alerting System

In this chapter the design and structure of a disaster and crisis alerting prototype system employing home automation systems is presented. It covers the architecture, prerequisites and technical aspects.

As already mentioned the scenario of a disaster alerting approach employing home automation systems consist of several components. The major task load will be on the two elements for interconnecting smart buildings and disaster alerting services. The Disaster Information Services (DIS) has a central position in this design acting as access point for the smart homes, which requests for disaster information. The control center is reachable over the internet. Furthermore, the control center provide a functionality for national authorities to collect or create incident information by themselves and provide this information via a public interface, which can be requested by the smart homes.

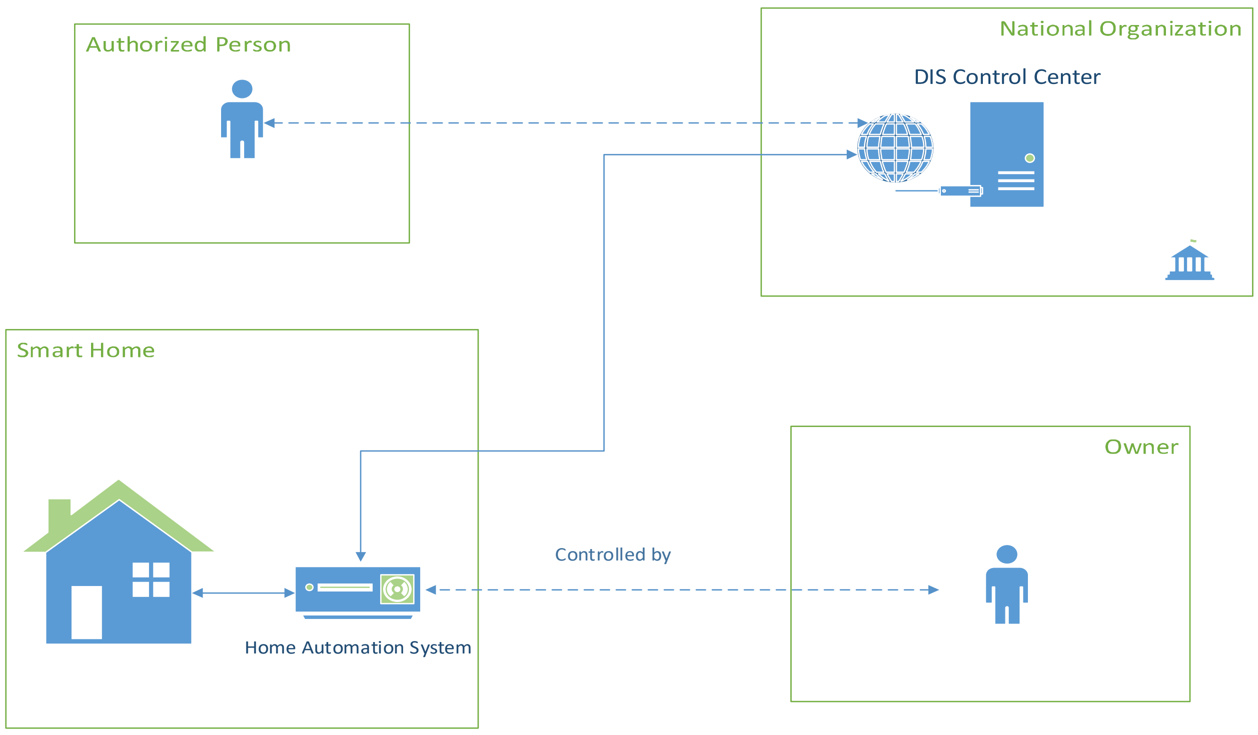


Figure 20: Abstract system architecture

The endpoint takes the second major part of the system. The DIS endpoint is connected with the DIS service and the home automation system over a secure internet connection. It is the interface between the home automation system and the disaster information service. The endpoint/middleware is designed to transform and forward risk mitigation events within the home automation bus.

3.1. Disaster Information Service Architecture

The basic architecture of the disaster alert and response system is based on a multi-tier architecture. Multi-tier architectures allow a flexible setup of complex distributed systems based on the client-server principle. Basically, tier-architectures are built up in layers. The more complex the requirements the more layers are used. Such a modular design makes it very flexible to respond to future requirements and changes. The following figure shows the disaster information service internal architecture at one site.

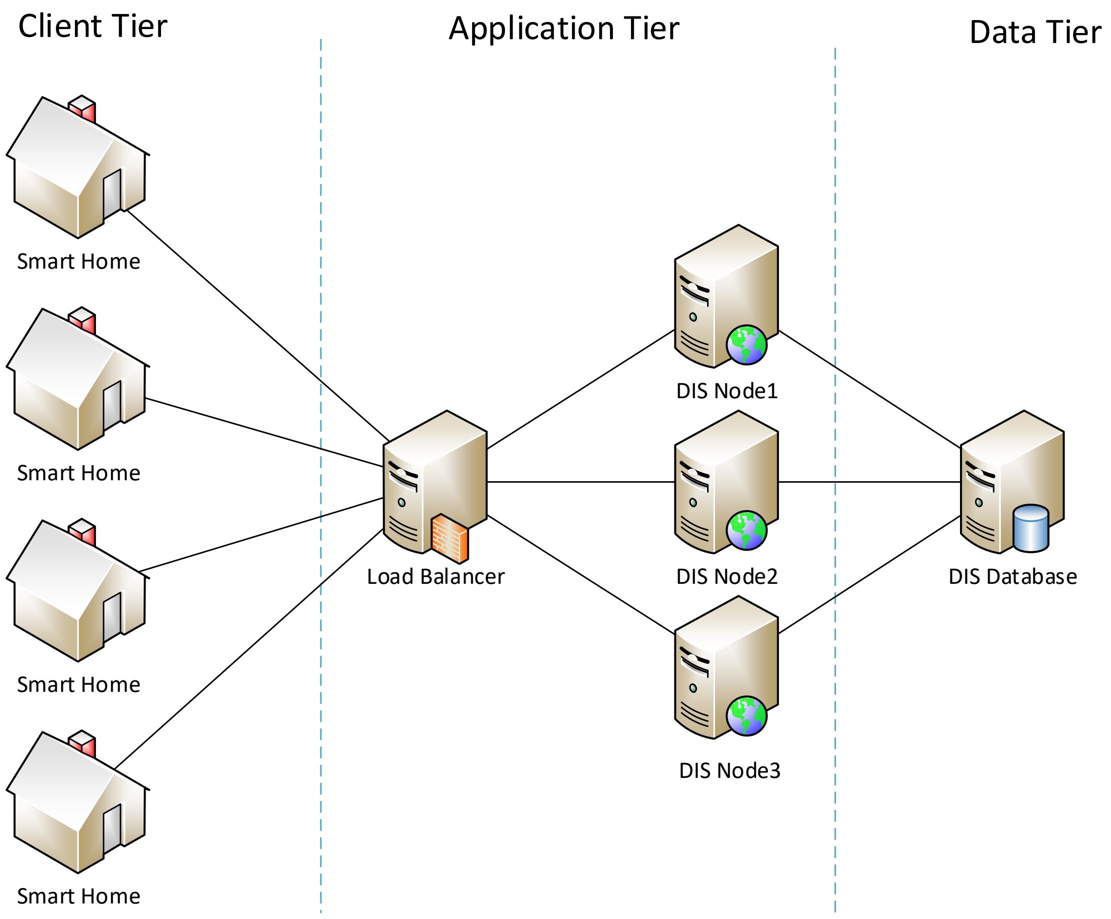


Figure 21: Backend architecture

As it is recommended, separation between application and data tier increases availability, flexibility and extensibility. Based on that design, the data at rest is stored at a separate layer. A central database allows consistent information storage. In addition, all information can be easily maintained by authorized people of a national organization. The disaster information service inside the application layer provides the information stored in the database. Because of the separation of application and data layer, it is possible to implement multiple nodes at one site for load balancing. An internet-facing load-balancer is used to forward the client requests to the nodes based on performance statistics of the disaster information service hosting nodes.

RESTful Web Service

In order to ensure a high availability of the entire system, it is necessary to choose a system architecture and communication strategy which optimally supports features such as scalability, reliability and elasticity. A wide spread and well-established architectural style for internet-based end-to-end communication is given by RESTful web service. Web services have gained tremendous importance in the age of internet-based communication, as it is possible to build lightweight and easy implementable APIs to provide a communication interface for all kind of devices.

The architecture style REST was first introduced in the year 2000 by Roy Thomas Fielding in his dissertation entitled "*Architectural styles and the design of network-based software Architectures*" [46]. REST stands for representational state transfer. It is an architectural style and not a methodology or a process. Fielding only abstract conditions, but do not specify explicit processes, protocols or even media. Typical architectures from the range of distributed applications based on certain characteristics were compared. On this basis, Fielding created an architecture pattern, with focus on scalability, extensibility and interoperability.

A RESTful API includes the following aspects [46]:

- Client-Server
- Stateless
- Caching
- Uniform Interface
- Layered System
- Code on Demand

By encapsulating the client and the server, they can be developed separately from each other, or the client can be kept more easily portable. The server is slimmer and supports the aspect of scalability. Stateless or statelessness does not mean that the client and server are not allowed to change their state. However, each operation must be completed in itself. REST dictates that the state is either held by the client or converted to a resource state by the server [46]. This has several advantages. The API is kept clear and does not have to determine from the context, what a request means. Links to resources can be reused easily. Statelessness also has a positive effect on stability and scalability in connection with caching, since no separate session data is required. The *caching* itself lowers the latency and the incoming data volume. With a layered system, modular design is required, which improves maintenance, reusability and extensibility. Code-on-demand is designed to provide greater client and server decoupling, expandability and scalability, as it can be deployed to the client as needed.

An important design aspect was the efficient use of web intermediaries (firewalls, proxies and caches) and the meaningful embedding of successful technologies such as HTTP, HyperText Markup Language (HTML) or XML and JSON.

As it is typical for emergency or disaster alerting, no access restrictions should be made for requesting clients. Critical information should be available for all. So there is no need of endpoint authentication and authorization. The information can be simply provided via a public interface, which the clients can access. This approach is comparable to public information on the web. Under this aspect, the HyperText Transfer Protocol (HTTP) in combination with TLS provides an easy solution supporting the client-server approach. In addition, most home automation systems have already a HTTP-Client implemented, so there minimal modifications necessary to establish an information channel.

3.2. Communication Flow and Data Exchange

A client-facing RESTful web service is hosted on the disaster information service nodes. For the information provision of incidents information, HTTP GET method have to be implemented. POST, PUT, DELETE methods are not required, because clients should only read information and should not be able to modify the data on the backend. This is only permitted to authorized people via an internal management interface.

In principle, resources can be requested via a specified URL (Uniform Resource Locator) by using the following schema: *`https://<server>:<port>/path`*. In a RESTful manner, each URL represents exactly one resource on the server. Furthermore, the pull-based approach makes it necessary to split the information about incidents into different parts for efficient use. Because sending all incident information to a client, even if it is not affected, will waste network bandwidth. Therefore the REST API has to implement two URLs where clients can request meta-information and more specific information of incidents.

HTTP-Method	**Purpose**
`GET /incidents/<region>`	Provides a list of incidents with meta-information, e.g. incident ID number and the affected area.
`GET /incident/<ID>`	Detailed information about the incident. For example, event type, urgency, instructions, etc.

Table 4: RESTful HTTP GET-Methods

The clients send request for new or modified incident meta-information periodically (e.g. every minute). The meta-information is a compact representation of all incidents affecting people and goods in a specific region. Primarily, this information is used by the endpoint first to decide if its location is affected and further request are necessary. If the endpoint is effected, comparing its geographic coordinates with the geo-coordinates by one of the listed incidents, the corresponding incident ID is used to send further requests in order to get detailed information about the current situation.

In addition to reduce the network traffic, it is meaningful to rank and categorize incident meta-information by their geographic regions, for example into states and cities. So, clients are able to request for information only they are interested in. For example, a client needs only information about current incidents in Upper Austria or Vienna, instead of receive the entire list of incidents in Austria.

The following figure is a simplified illustration of the communication flow:

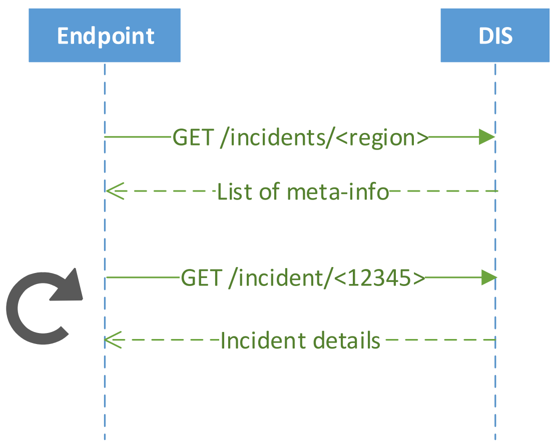


Figure 22: Information Request Sequence

This request sequence has the advantage that the network and server-load can be reduced by requesting meta-data at first. Detailed information is only requested by the client when it is actually needed.

3.3. Caching and Conditional Requests

In addition, HTTP caching mechanisms can be used to increase the response time and reduce network and server-side load. The basic idea of HTTP caches is to store resources locally for faster retrieval the next time a resource or an object is requested [47]. This can also be applied in context with pull-based disaster information alerting system, as shown in the figure below.

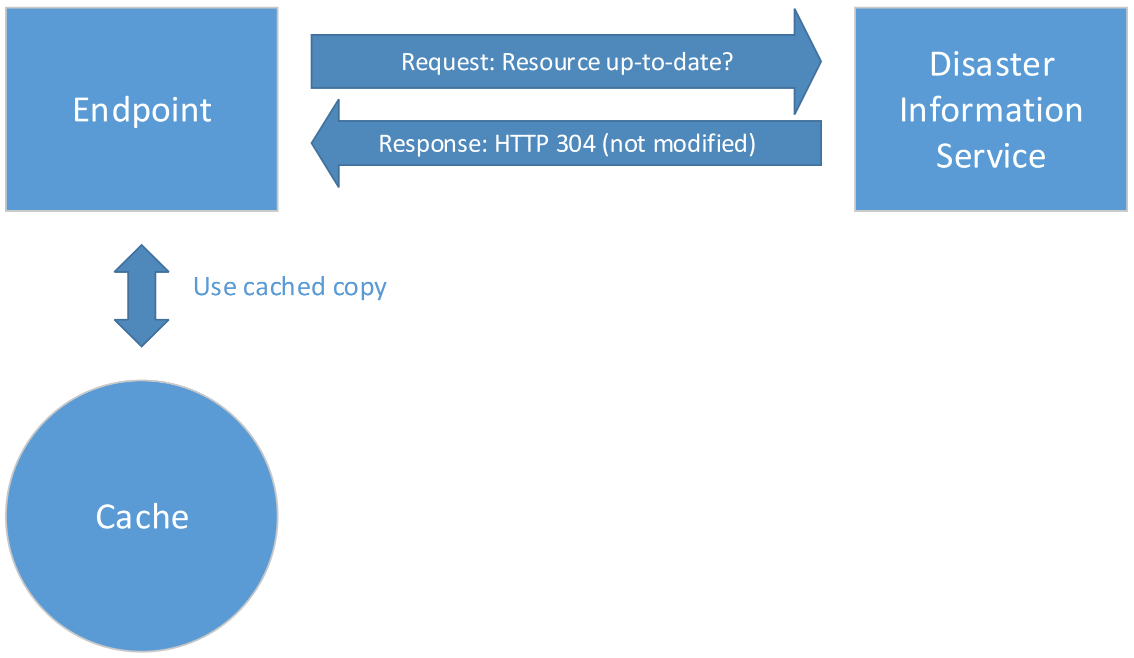
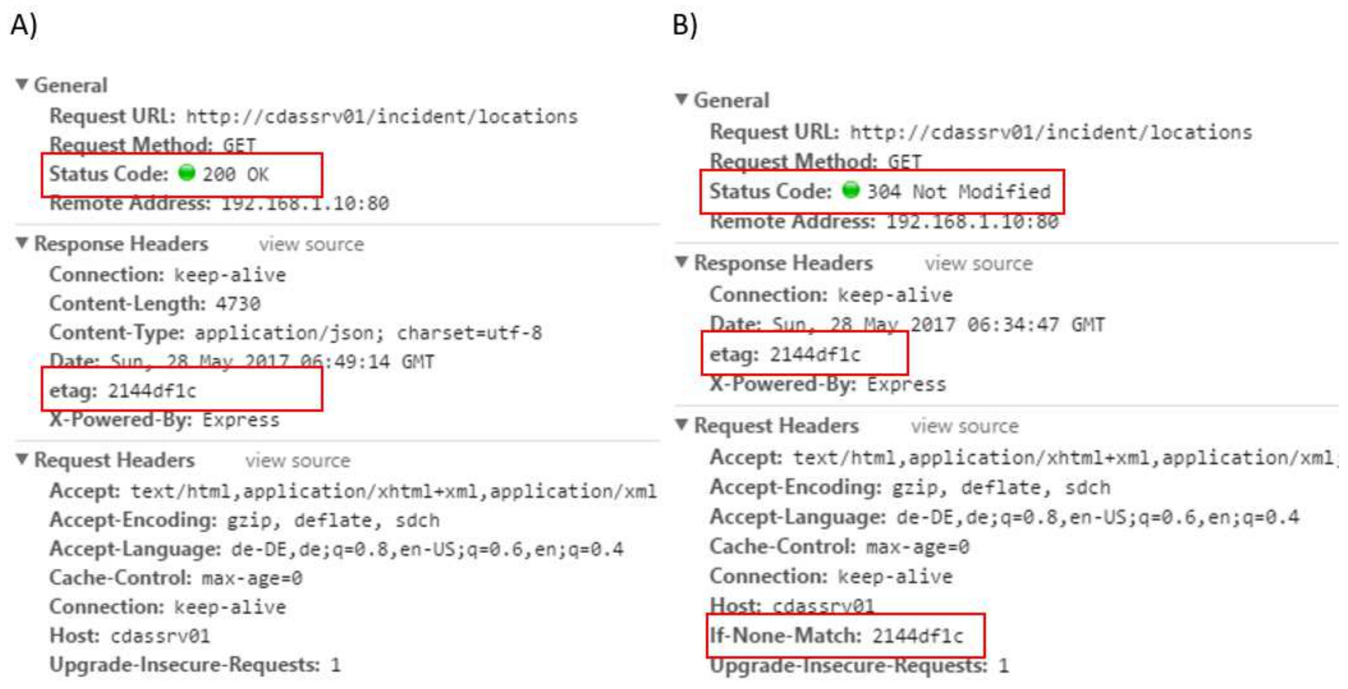

Endpoint
Request: Resource up-to-date?
Response: HTTP 304 (not modified)
Disaster Information Service
Use cached copy
Cache

Figure 23: HTTP caching mechanism [47]

The conditional request is used to ask the disaster information service if it has a modified copy of a resource. The client send information about the cached object in the HTTP-Header of the request. The disaster information service will determine, based on the header information, if the client cached incident version is the most recent. If so, the disaster information service respond with the HTTP status code 304 and an empty response body, which means that the resource has not changed since the last request. In the case, the incident information is outdated, the disaster information service return an update.

To fulfill this task, so called Entity-Tag (ETag) Header information has to be included in the request header sent by the client. The ETag represents a digest of the resources contents, for instance, an MD5 hash. The first time, the client requests a resource from the service, an ETag is included in the response header from the server. This value is used in the IF-None-Match request header for upcoming requests for identification, if the resource at the client-side is the most recent version.

The following figure demonstrates the caching mechanism, implemented in the disaster information service:



A)

```
▼ General
    Request URL: http://cdassrv01/incident/locations
    Request Method: GET
    Status Code: ● 200 OK
    Remote Address: 192.168.1.10:80
▼ Response Headers      view source
    Connection: keep-alive
    Content-Length: 4730
    Content-Type: application/json; charset=utf-8
    Date: Sun, 28 May 2017 06:49:14 GMT
    etag: 2144df1c
    X-Powered-By: Express
▼ Request Headers      view source
    Accept: text/html,application/xhtml+xml,application/xml
    Accept-Encoding: gzip, deflate, sdch
    Accept-Language: de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4
    Cache-Control: max-age=0
    Connection: keep-alive
    Host: cdassrv01
    Upgrade-Insecure-Requests: 1
```

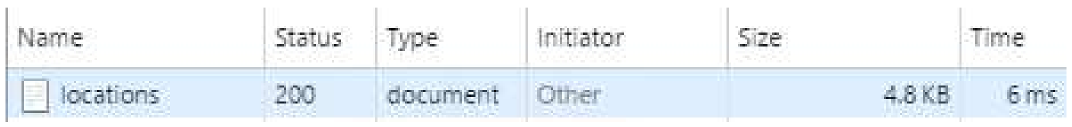
B)

```
▼ General
    Request URL: http://cdassrv01/incident/locations
    Request Method: GET
    Status Code: ● 304 Not Modified
    Remote Address: 192.168.1.10:80
▼ Response Headers      view source
    Connection: keep-alive
    Date: Sun, 28 May 2017 06:34:47 GMT
    etag: 2144df1c
    X-Powered-By: Express
▼ Request Headers      view source
    Accept: text/html,application/xhtml+xml,application/xml;
    Accept-Encoding: gzip, deflate, sdch
    Accept-Language: de-DE,de;q=0.8,en-US;q=0.6,en;q=0.4
    Cache-Control: max-age=0
    Connection: keep-alive
    Host: cdassrv01
    If-None-Match: 2144df1c
    Upgrade-Insecure-Requests: 1
```

Figure 24: HTTP Conditional Requests

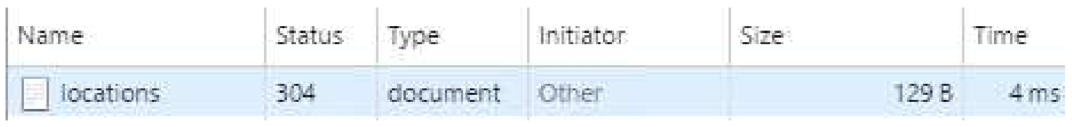
As it can be seen in the first request (A), the server response contains the ETag value and the entire requested resource, with a content-length of 4730 bytes. In a second request (B), the resource at the server-side has not been modified. Therefore, the response contains header information only, without any resource data.

Based on this behavior, this allows to reduce the response size and network load significantly, as shown in the figures at the next page.



Name	Status	Type	Initiator	Size	Time
locations	200	document	Other	4.8 KB	6 ms

Figure 25: HTTP 200 response size



Name	Status	Type	Initiator	Size	Time
locations	304	document	Other	129 B	4 ms

Figure 26: HTTP 304 response size

In consumption, conditional HTTP request can significantly save network bandwidth and transferring costs. At this point, it has to be noted that the response size depends on the transferred information. The more incident information is contained, the bigger is the request size for sure.

3.4. Load Balancing Strategy

With an increasing number of request the disaster information service can quickly reach performance limits which results in bad response time or even worse service outages. In order to create a scalable disaster information service system, load-balancing strategies have to be considered.

The following figure illustrates the load-balancing strategy for the envisioned system:

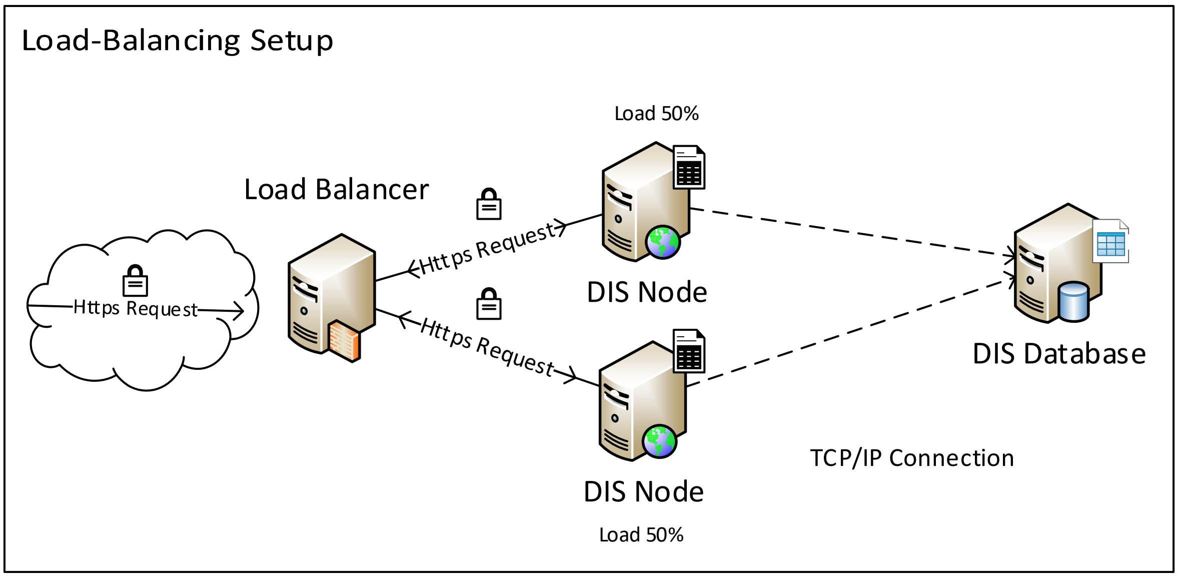


Figure 27: Load balancing configuration

Instead of using only one web-service node to answer client requests, it is more efficient in respect to performance capabilities to build a cluster of disaster information service nodes to be able to share the load. In this illustration, the load balancer acting as a HTTPS proxy forwarding requests to several disaster information service nodes based on decision rules (e.g. round-robin or number of connections per node). This setup allows the flexibility to add new nodes and ensure performance of the disaster information service.

3.5. Data Structure

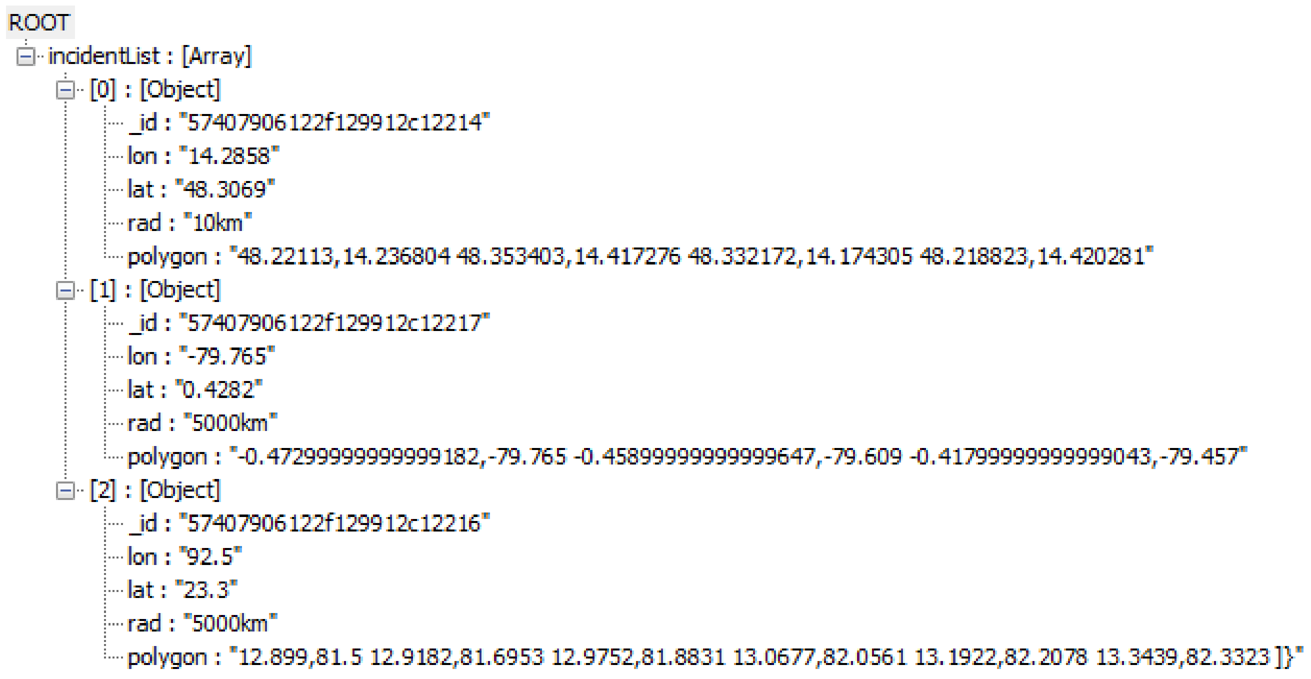
In order to enable endpoints to identify whether they are affected by an incident, an appropriate data exchange format is required. At this point, geographical information is an important factor. Each geo-information per incident provided to the clients consists of the following values.

Key	Value
`id`	Unique identifier of an incident
`lon`	Longitude of an incident (based on GPS coordinates)
`lat`	Latitude of an incident (based on GPS coordinates)
`rad`	Radius of the incidents sphere of influence
`polygon`	Optional: List of latitude and longitude coordinates

Table 5: Incident meta-info

The `id` is a unique value to identify the incident, used for specific request to receive detailed information. Longitude (`lon`) and latitude (`lat`) describe the center of incident occurrence. `rad` and `polygon` values are required to define the area of influence. Whereas the mandatory radius contains a distance from the center of the incident occurrence to define a rough area, the optional polygon field can be used to specify an exact area based on geographical coordinates.

The following data structure is a lists of incidents, which contains the mentioned meta-information:

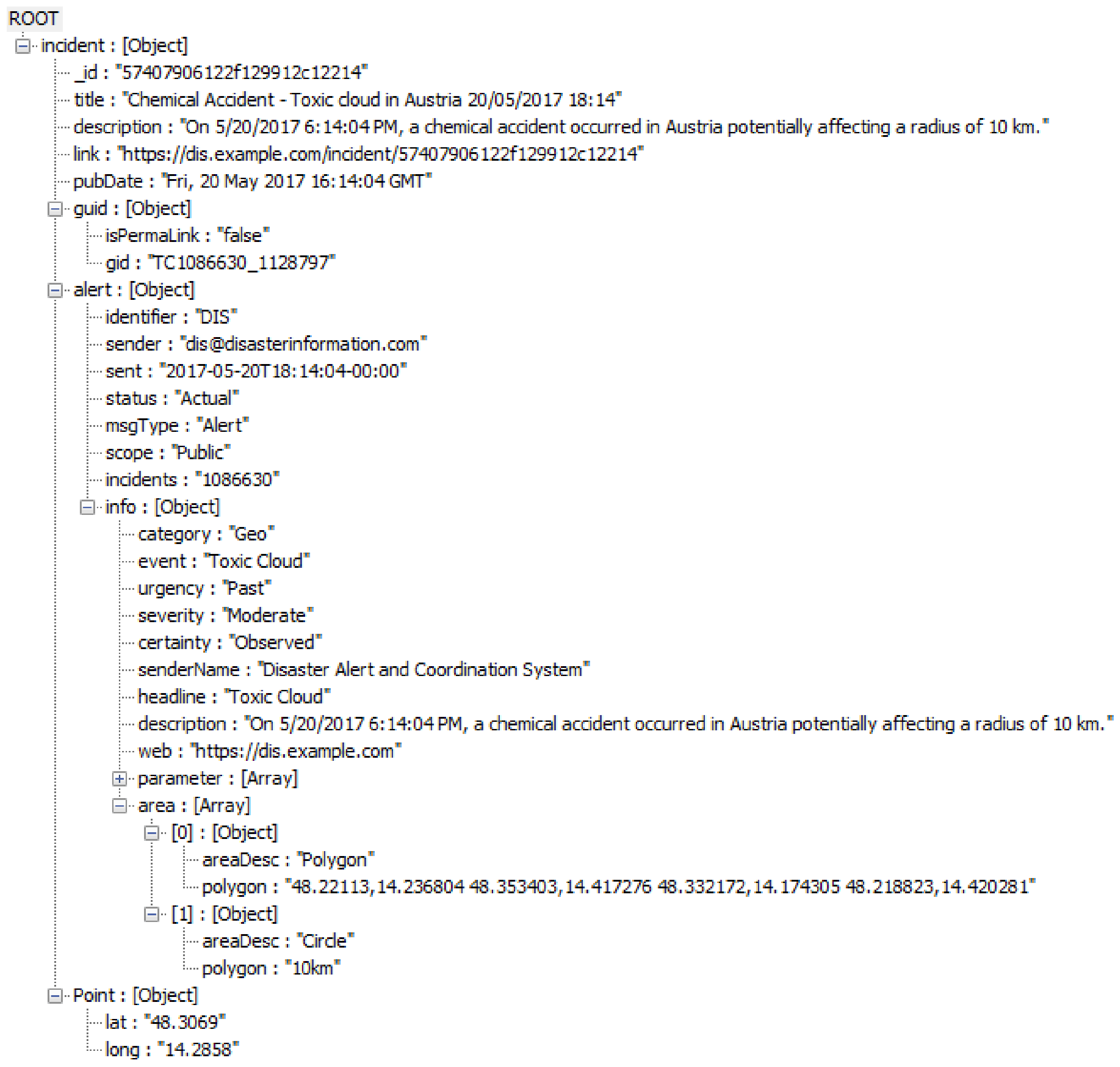


```
ROOT
incidentList : [Array]
    [0] : [Object]
        _id : "57407906122f129912c12214"
        lon : "14.2858"
        lat : "48.3069"
        rad : "10km"
        polygon : "48.22113,14.236804 48.353403,14.417276 48.332172,14.174305 48.218823,14.420281"
    [1] : [Object]
        _id : "57407906122f129912c12217"
        lon : "-79.765"
        lat : "0.4282"
        rad : "5000km"
        polygon : "-0.47299999999999182,-79.765 -0.45899999999999647,-79.609 -0.41799999999999043,-79.457"
    [2] : [Object]
        _id : "57407906122f129912c12216"
        lon : "92.5"
        lat : "23.3"
        rad : "5000km"
        polygon : "12.899,81.5 12.9182,81.6953 12.9752,81.8831 13.0677,82.0561 13.1922,82.2078 13.3439,82.3323 ]}"
```

Figure 28: Incident meta-information in JSON format

The data can be received using the predefined URL */incidents/<region>*. The format of an incident alert is inspired by CAP-Standard and structured with the JavaScript Object Notation (JSON). JSON provides an efficient way to structure information and is easier to read for humans and for machine-to-machine communication [48].

A second client request using the URL pattern `/incident/<id>` will return details of a specific incident, as shown in the figure below:



```
ROOT
⊟ incident : [Object]
    _id : "57407906122f129912c12214"
    title : "Chemical Accident - Toxic cloud in Austria 20/05/2017 18:14"
    description : "On 5/20/2017 6:14:04 PM, a chemical accident occurred in Austria potentially affecting a radius of 10 km."
    link : "https://dis.example.com/incident/57407906122f129912c12214"
    pubDate : "Fri, 20 May 2017 16:14:04 GMT"
  ⊟ guid : [Object]
      isPermaLink : "false"
      gid : "TC1086630_1128797"
  ⊟ alert : [Object]
      identifier : "DIS"
      sender : "dis@disasterinformation.com"
      sent : "2017-05-20T18:14:04-00:00"
      status : "Actual"
      msgType : "Alert"
      scope : "Public"
      incidents : "1086630"
    ⊟ info : [Object]
        category : "Geo"
        event : "Toxic Cloud"
        urgency : "Past"
        severity : "Moderate"
        certainty : "Observed"
        senderName : "Disaster Alert and Coordination System"
        headline : "Toxic Cloud"
        description : "On 5/20/2017 6:14:04 PM, a chemical accident occurred in Austria potentially affecting a radius of 10 km."
        web : "https://dis.example.com"
      ⊞ parameter : [Array]
      ⊟ area : [Array]
        ⊟ [0] : [Object]
            areaDesc : "Polygon"
            polygon : "48.22113,14.236804 48.353403,14.417276 48.332172,14.174305 48.218823,14.420281"
        ⊟ [1] : [Object]
            areaDesc : "Circle"
            polygon : "10km"
  ⊟ Point : [Object]
      lat : "48.3069"
      long : "14.2858"
```

Figure 29: Details of an incident in JSON format

The JSON object which contains detailed incident information is related to the data format as given by the CAP standard, described in section 2.2. This data package includes the most important information which is needed to alert people via smart homes about:

- What has happened,
- Where has it happened,
- Who is affected,
- Which risk mitigation actions should be taken to protect yourself

This data object will be used by home automation systems for information visualization and can also include recommendations for self-protection.

3.6. Peer-to-Peer Communication Capabilities

As discussed in section 2.4, sharing information can increase scalability and lower possible performance issues at the server-side. In addition to the centralized approach using a web-service and client requests, a peer-to-peer communication approach is going to be introduced. The following concept can be seen as extension/alternative to the existing approach.

Challenges in peer-to-peer communication are higher than in centralized client/server approaches, but it in terms of scalability and availability by increasing number of participants more efficient. The following figure should illustrate a concept using a peer-to-peer communication techniques in context with a disaster alerting system:

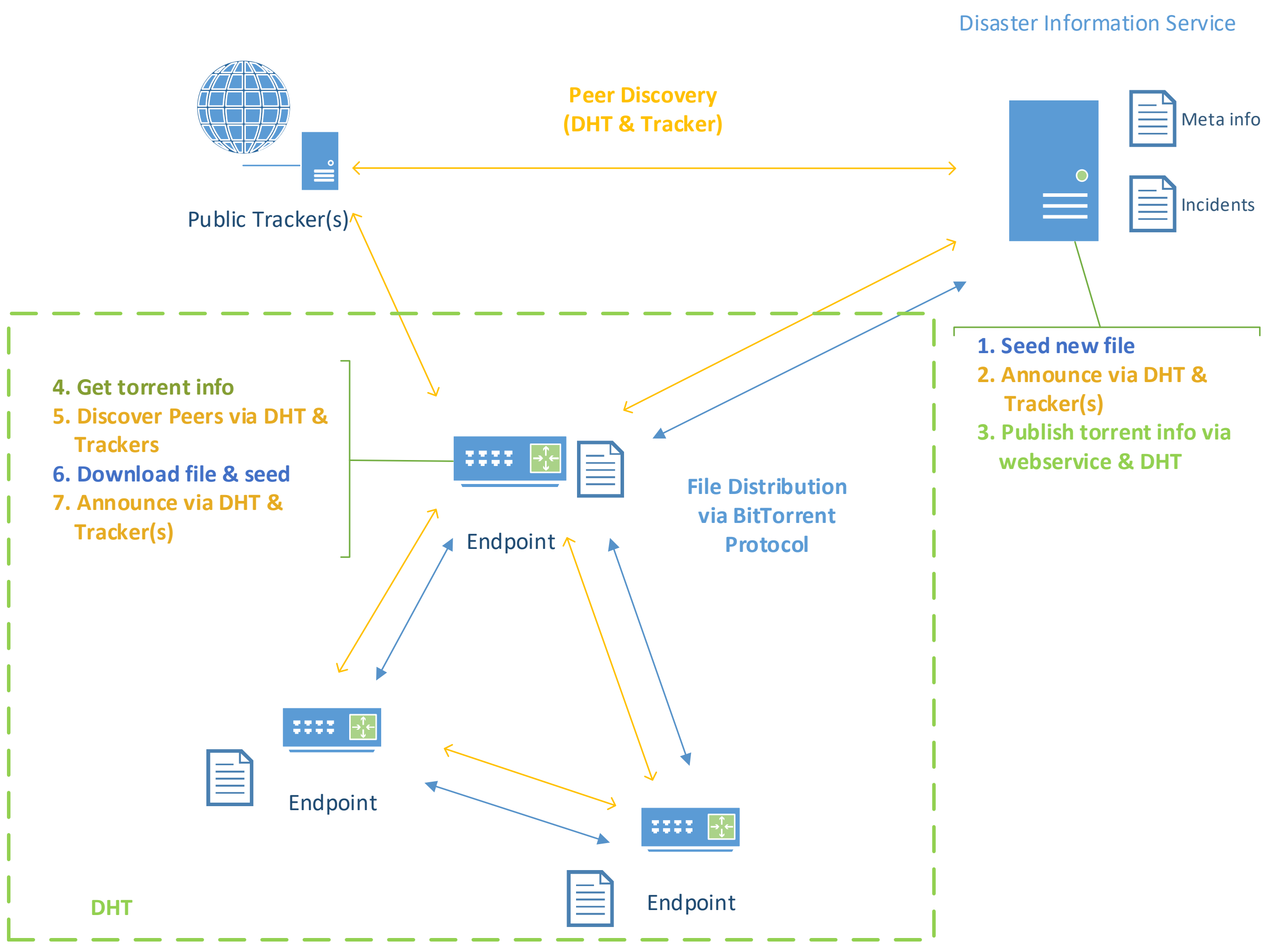

Disaster Information Service
Peer Discovery
(DHT & Tracker)
Public Tracker(s)
Meta info
Incidents
1. Seed new file
2. Announce via DHT & Tracker(s)
3. Publish torrent info via webservice & DHT
4. Get torrent info
5. Discover Peers via DHT & Trackers
6. Download file & seed
7. Announce via DHT & Tracker(s)
Endpoint
File Distribution via BitTorrent Protocol
Endpoint
Endpoint
DHT

Figure 30: Peer-To-Peer Information Distribution

As illustrated in Figure 30, a central disaster information service plays a vital role in a peer-to-peer information sharing approach. The central disaster information service acts as initial seeder. Its task is to share incident information (meta-info and incident details, as already mentioned) with endpoints interested in. The important part here is that only one master node is allowed initially to create torrent meta-info and seed incident information. These meta-info (for example in form of magnet links) is required to enable participating peers to find and download related incident information distributed. But first, the peer have to gather this meta-info from somewhere. In addition, incident information can be changed which makes it necessary to frequently update this meta-info. One way is, to use the introduced web-service in order to build a contact point for the peers and publish these magnet links and as mutable DHT items. Hence, we have to come back to some sort of centralization. Peers request for these torrent meta-info frequently. Based on this information they can start a discovery processes for seeding clients based on DHT and/or utilizing Trackers for signaling peer information. The downloaded file has to be announced and seeded to others in order to achieve full peer-to-peer capabilities.

3.6.1. Information distribution

In contrast to the central client-server approach where meta-information about incidents is published on the web-service, frequently requested by endpoints to detect information changes, peer-to-peer capabilities (BitTorrent DHT) can be used to publish and notify endpoints about changes to incident meta-information. But first, contact information have to be published. New participating endpoints have to initially request for contact information, which is associated with the region of interest. Therefore, it is required by the central disaster information service to publish these contact information first. The association is represented in Figure 31.

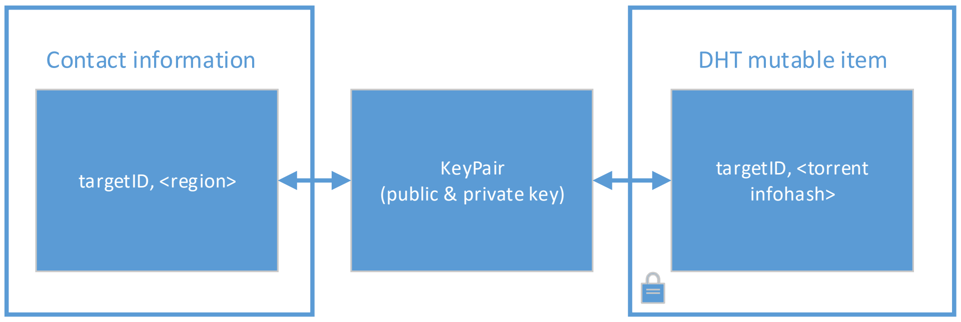


Figure 31: Contact information of region

The region of interest is associated with a key pair. In principle, the private key is used by the disaster information service to sign mutable DHT items, whereas the public key is used for verification purposes for DHT storage operations and simultaneously as identifier for a region and DHT mutable item. So endpoints use the published contact info on the web-service in order to identify mutable items via the targetID in the DHT network.

Publishing incident meta-information

Now, the endpoint knows under which ID he can find and request incident information in the DHT network. In a next step, the disaster information service is responsible for seeding incident meta-information via BitTorrent protocol and store the signed torrent infohash in the DHT network. The publishing process initiated by the central disaster information service can be described as follows:

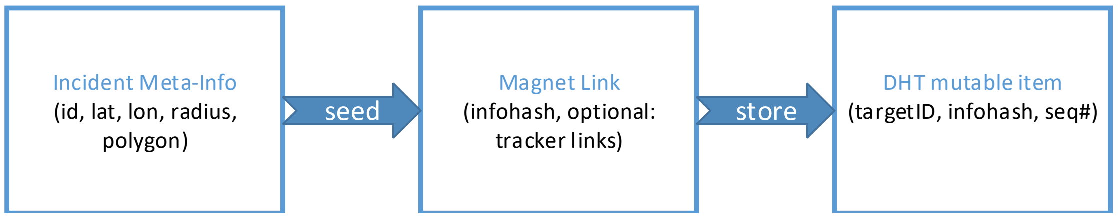

Incident Meta-Info
(id, lat, lon, radius, polygon)
seed
Magnet Link
(infohash, optional: tracker links)
store
DHT mutable item
(targetID, infohash, seq#)

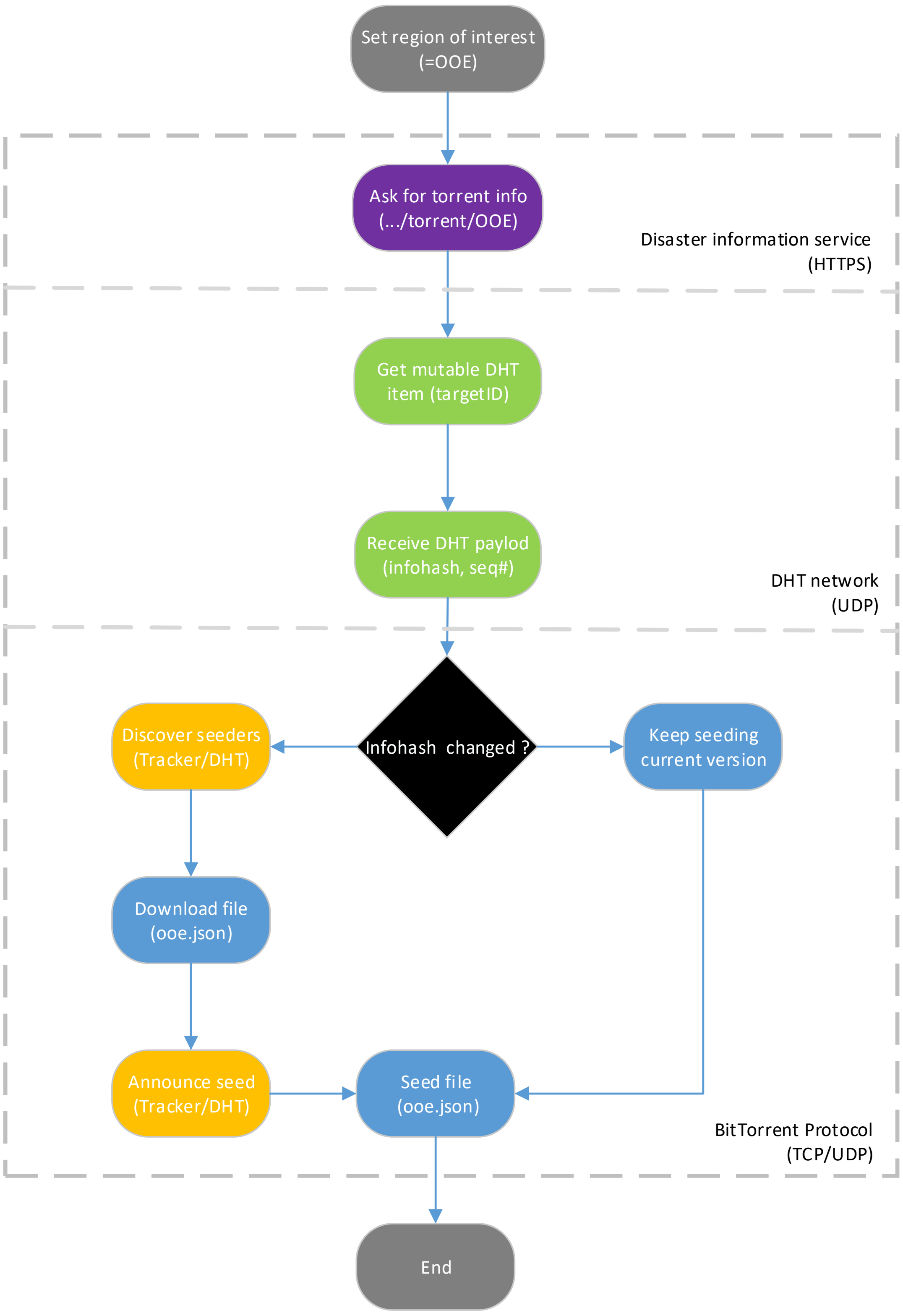
Figure 32: Information distribution and publishing

Based on current incident information, a torrent is created containing the infohash and optionally tracker links. These information is required to enable endpoints do find and download incident information from seeding peers. In addition, a mutable DHT item is created, based on torrent’s infohash signed with the private key and a sequence number of the payload. The sequence number is increased each time when the disaster information service stores/update new content in the DHT network. So a node in the DHT network receiving a storage request compares the current sequence number with the new one. If the sequence number is higher than the current one, the item is accepted. Otherwise, the request will be discarded by the storing node.

Only for clarification purposes, transmission of incident data (incident meta-info) is separated by using the BitTorrent protocol, whereas the process for incident update notification (based on torrent infohash) is done via the DHT network. This separation is required because the storage space of DHT items is limited to 1000 byte.

3.6.2. Information consumption and sharing

In a first step, if an endpoint want to receive incident information, the region of interest has to be configured, for example OOE. Every time the region has been changed, the endpoint have to request the initial torrent information (e.g. targetID) associated with the region. Therefore, it sends a request to the disaster information service hosting the contact information. These information is the basis for checking updates to the torrent meta-info and will not change. So every endpoint interested in region OOE will use exactly the same information to check if something has changed. The payload of the DHT mutable item contains the latest infohash published through the disaster information service in the DHT network. After the endpoint knows the latest infohash published, it compares the current one with the infohash received.



Set region of interest (=OOE)
Ask for torrent info (.../torrent/OOE)
Disaster information service (HTTPS)
Get mutable DHT item (targetID)
Receive DHT paylod (infohash, seq#)
DHT network (UDP)
Infohash changed ?
Discover seeders (Tracker/DHT)
Keep seeding current version
Download file (ooe.json)
Announce seed (Tracker/DHT)
Seed file (ooe.json)
BitTorrent Protocol (TCP/UDP)
End

Figure 33: Peer-to-Peer incident information consumption

If torrents meta-info has changed, then a download sequence is initialized by discovery endpoints who have already seeding the latest file containing incident information. In addition, remove the current file from the local seeding repository and start seeding the latest version. Otherwise, if the infohash is the same, keep seeding the current file. Based on a time interval (e.g. 1 minute) start the lookup process again and verify if incident updates are available.

This peer-to-peer communication approach guarantees that every endpoint can download and sharing the correct information with others. The more endpoints interested in a specific region the more contact points are available where this information can be gathered.

3.7. Request Scheduling

The design of the disaster information service also affects the design of the endpoint/middleware. Different communication strategies have been discussed how the communication flow & information can be realized. Both approaches are based on a pull-method. Therefore, endpoints are urged to frequently send information request, compare the results and check if its location is affected by incidents based on the geographical coordinates.

As it is commonly known an emergency warning system can only fulfill its purpose when incident information is provided and distributed promptly. The disaster information system design requires that endpoints frequently ask for new information. To handle this load on the server-side, scalability and performance aspects are already considered in the design for the disaster information service. But indeed precautions can be made on the client-side, e.g. using an optimized client request strategy. Beside the fact that the web-service has to handle concurrent requests anyway, a fixed request time-interval set on the client-side could provide the possibility that different located clients send request exactly at the same time. And that could have a negative impact on performance as well. In order to prevent this possibility it is recommended to vary the time when requests are sent.

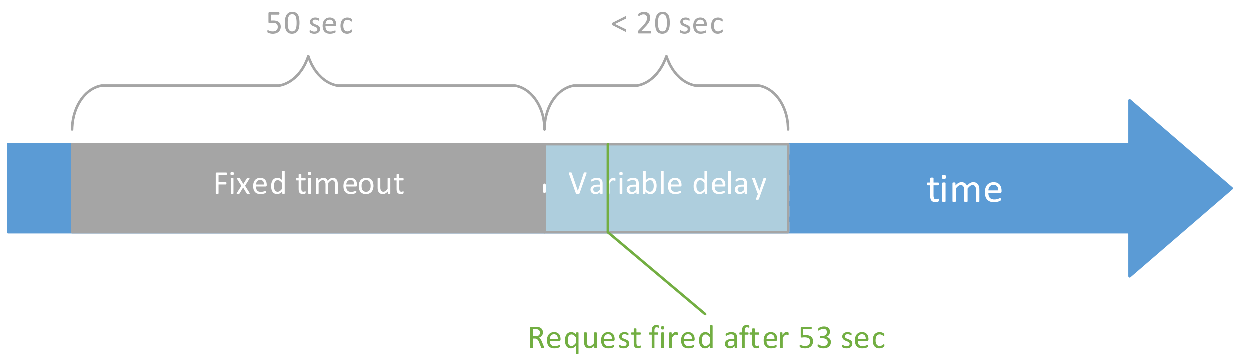


Figure 34: Scheduling Client Requests

As shown in Figure 34, a fix timeout is set which initiates the client request procedure for example every 50 seconds. In addition, a delay range can be used, for example between 0 and 20 seconds, to calculate a random value which is added to the interval. After a request is sent to the disaster information service based on this value, the client waits another 50 second + the random value calculated to perform the next request. This mechanism should prevent concurrency issues made by software design.

3.8. Incident evaluation and calculation

Incident meta-information include a geographical coordinates (latitude and longitude) of the incident and diameter information. This information is used to calculate if the location of the endpoint (based on its latitude and longitude) is affected. To calculate a distance between two coordinates the Haversine formula [49] has been used. Based on this formula it is possible to find the shortest path on Earth's surface by latitude and longitude.

Haversine function: $$\text{haversine}\ \theta = \sin^2\left(\frac{\theta}{2}\right)$$

Formula 2: Haversine function [49, p. 4]

To solve for d using inverse sine function:

$$d = 2\mathrm{r}\sin^{-1}\left(\sqrt{\sin^2\left(\frac{\varphi 2 - \varphi 1}{2}\right) + \cos(\varphi 1)\cos(\varphi 2)\sin^2\left(\frac{\lambda 2 - \lambda 1}{2}\right)}\right)$$

Formula 3: Equation for distance calculation [49, p. 4]

(d) is the calculated distance between two points based on latitude (φ) and longitude (λ). (r) is the radius of the sphere [49, p. 4].

As the distance between the endpoint and the occurrence of the incident is known it is simple to determine if the endpoint is affected by the incident. If the calculated distance (d) is less or equal than the given radius (radius) of the incident, then the endpoint affected.

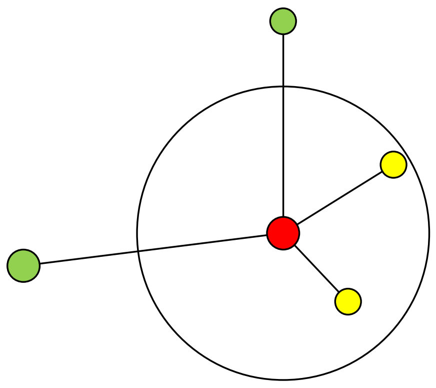


Figure 35: Calculation of distance between two points

At this point it has to be noted that the Earth is not perfect shaped. The Earth radius varies at the pole and equator. Therefore it is recommended to use the median Earth radius (6371 kilometers) for the calculation with Haversine formula to limit the flattering. Nevertheless, this can influence the result especially on long distances calculations. Because distance calculation in our case is limited to a limited area, the Haversine formula has been used.

4. Implementation of the Prototype

This section deals with the implementation of the prototype. Essential techniques and core functionality are highlighted. In addition, the development environment and software are described which are used to setup and create the system.

4.1. Backend-Database

According to the conceptual design, a backend-database is necessary to store and manage incident information. For this prototype a NO-SQL database named MongoDB is used. MongoDB is one of the market-leading NoSQL databases. MongoDB [50] was designed for modern IT landscape and allows an agile development and higher scalability of applications. MongoDB is a general-purpose open-source database.

MongoDB provide the following key features:

- Document data model with dynamic schemes
- Comprehensive, flexible index support and powerful queries
- Auto-Sharding for horizontal scalability
- Built-in replication for high availability
- Text search
- Enhanced security (e.g., Kerberos support)

Beside the scalability aspects, the main reason for MongoDB is because incident information can be stored in simply JSON format using the MongoDB document data model. The big advantage is that incident data remains in the same data structure as it is transmitted to the clients, which makes is very efficient to store, maintain and load data from the database. This result in positive performance effects, because no extra algorithm is required to parse data form the database into the right transmission format. Two data collections (`incident` and `users`) have been created to store incident and user information. While incident information is provided to the endpoints, user credentials are only required for user authentication from the incident management user interface.

To connect the disaster information service nodes with the database a middleware has been created in order to create, delete, and retrieve incident information.

Method	Description
`storeIncidentInDATABASE:function(incident, callback) {}`	Store incident in the database
`getAllIncidentsFromDATABASE : function(callback) {}`	Load all existing incident from database
`getIncidentFromDATABASE : function(id, callback) {}`	Get incident from database by incident ID
`deleteIncidentFromDATABASE : function(id, callback) {}`	Delete incident by incident ID

Table 6: MongoDB middleware

4.2. Disaster Information Service Node

The interacting components, e.g. the RESTful web service and also the endpoint/middleware are developed with Node.js and express.js as web-framework. Node.js [51] is an JavaScript runtime built on the JavaScript V8 engine of Chrome. It uses an event-driven, non-blocking input/output model which makes Node.js lightweight and efficient. It also provides a comprehensive open source repository (Node Package Manager [52]) that enables the integration of many modules and libraries. The asynchronous event driven JavaScript runtime is designed to create scalable network applications. Because of this fact many connection can be handled concurrently, in contrast to models following thread-based networking, which is relatively inefficient and difficult to implement.

Generally, in order to build a vertical and more horizontal scalable system, the RESTful disaster information service has a modular design and can run on several hosts without the need of changes to the configuration and code. That means it is possible to create a cluster of DIS nodes to increase performance if required.

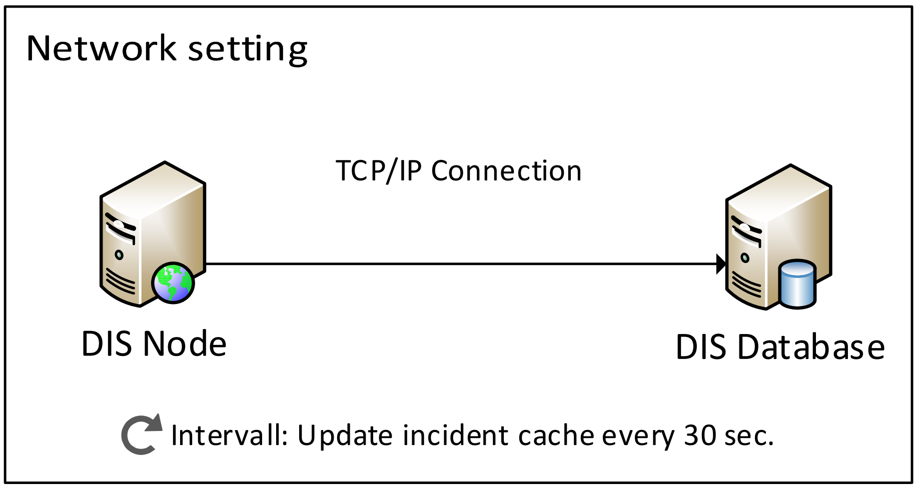


Figure 36: DIS information update

The disaster and information service nodes acting as an external content delivery node and update and cache incident information from the database on a regular basis. This modular implementation strategy should prevent high loads at backend-side and can be seen as content delivery node at one site.

The following code snipped illustrates how meta-information is generated based on the incident information:

```
function reloadIncidentGeoInfo(){
   …
   DBUtil.getAllIncidentsFromDATABASE(function (incidents) {
       incidents.forEach(function (elem) {
             if (elem.alert) {
                 var geolocation = {
                      "_id" : elem._id, “lon" : elem.Point.long, "lat": elem.Point.lat,"
                      rad" : elem.rad, "polygon" : elem.alert.info.area.polygon };
                      pubIncidentListCache.push(geolocation);
                   }});
              lastETag = crc.crc32(pubIncidentListCache).toString(16);
            });
        }
```

Listing 2: Load incident meta-info from DB

This incident-meta information is the provided to the endpoints. Based on the geo-information e.g. longitude and latitude, the endpoint can check whether it is affected by an incident or not.

The disaster information service API can externally accessed via GET requests:

HTTP GET Methods	Description
`app.get('/incidentList', function (req, res) {}`	Route to provide a meta-info of all incidents
`app.get('/incidents/OOE', function (req, res) {}`	Route to provide meta-info of incidents for a specific region
`app.get('/incident/:id', function (req, res) {}`	Route to get detailed incident information by incident ID.

Table 7: Disaster information service HTTP GET routes

As mentioned in the conceptual design, it is possible to request meta-information and details of incidents via HTTP GET requests. An important part of this implementation is only to respond with incident data if the requested information has been changed. The implementation logic is illustrated in the following example:

```
app.get('/incidents/OOE', function (req, res) {
                    if(req.headers.etag === lastETag){
                        res.setHeader('etag', lastETag);
                        res.status(304).end('');
                    }else{
                        res.setHeader('etag', lastETag);
                        res.send({incidentList : pubIncidentListCache});
                     }
});
```

Listing 3: Request handling based on ETags

Each time the endpoint send a request, the generated ETAG hash (calculated based on available incident information) is set in the response header. The next time an endpoint requests the same source (sending the ETAG as “`If-Non-Match`” in the request header) the info hash is compared to the current one. If both values match, the disaster information service responds with a HTTP status code `304 Not`

`Modified`. On the other hand, if the values are not the same the service will send the updated incident information.

4.3. Load-Balancer Configuration

In this setting, the NGINX web server works as load-balancer in front of the disaster information service nodes. NGINX [53] is an open source and highly scalable HTTP web server; reverse proxy and load balancer for HTTP, TCP and UDP traffic. NGINX is known for its high performance, stability, rich feature set, simple configuration, and low resource consumption.

The disaster information service listening on TCP/IP port *44331* for the internal communication with the load balancer. According to NGINX documentation, the ports for internal upstream communication must be different, because the external HTTPS requests terminate already at TCP/IP port *443* at the load balancer. The load-balancing mode is set to *least-connected*. Which means an incoming request is forwarded to the server with the least number of active connections.

The implementation of the load-balancer configuration is given below:

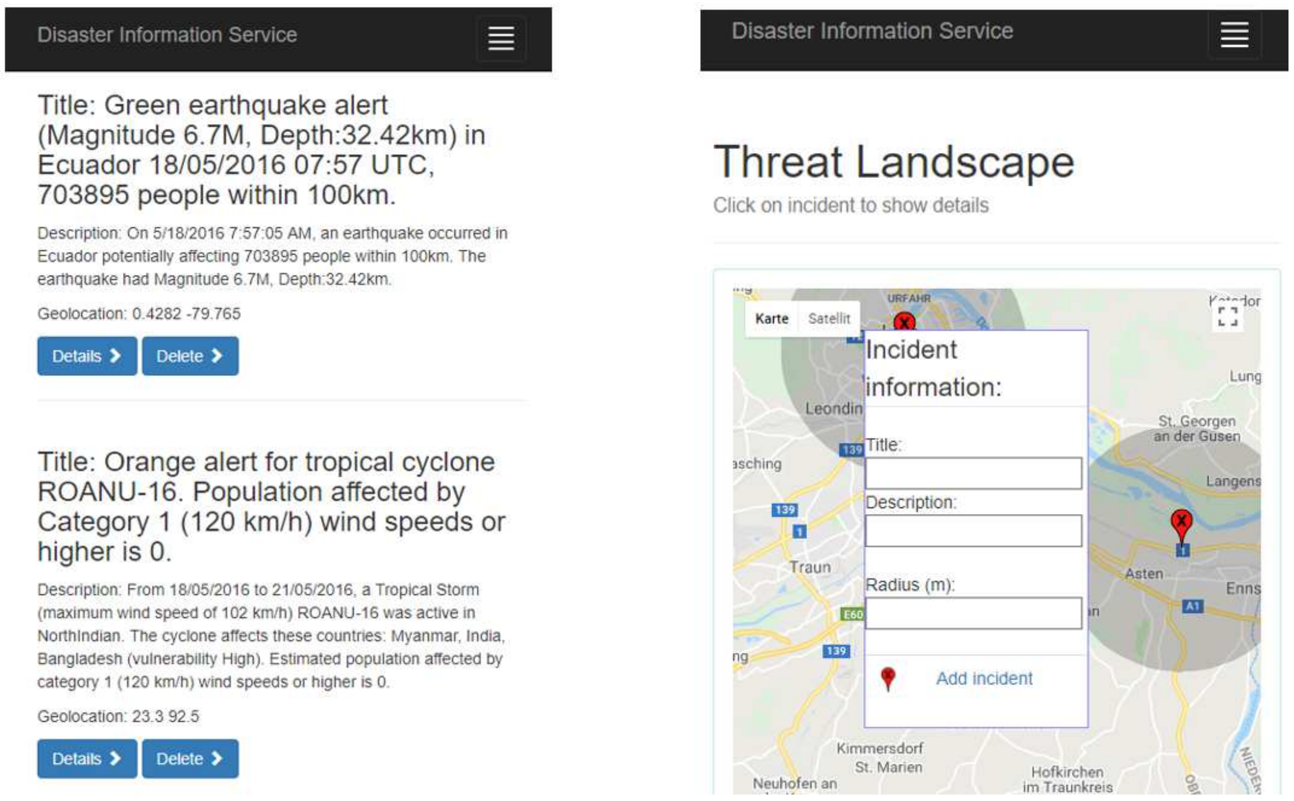
```
upstream cdncluster {
    least_conn;                 # Use Least Connections strategy
    server 192.168.1.10:4431;      # DIS NodeJS Server 1
    server 192.168.1.20:4431;      # DIS NodeJS Server 2
}
server {
    listen 443 ssl;
    server_name dis.example.at; ssl on;
    ssl_certificate         /etc/nginx/cdn/fullchain.pem;
    ssl_certificate_key     /etc/nginx/cdn/privkey.pem;

 location / {
        proxy_set_header Host $host; proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_pass https://cdncluster; proxy_redirect off;
    }
}
```

Listing 4: NGINX load balancer configuration

As it can be seen in the configuration file, the load balancer configuration handles request for two disaster information service nodes. In this example, only encrypted connections are allowed to be forwarded to the participating server nodes at the back. As stated by Nelson [54], it can be useful to offload SSL/TLS decryption from the backend nodes, because SSL/TLS decryption is CPU intensive and can lower performance of the content delivery nodes.

4.1. Disaster Information Service - Management Interface

For simulation purposes and to manage incidents, e.g. create or delete, a web-based management console has been developed. It can be seen as stand-alone server which is connected to the database. The feature set includes an overview of all incidents shown in a map and simple list.


Disaster Information Service

Title: Green earthquake alert (Magnitude 6.7M, Depth:32.42km) in Ecuador 18/05/2016 07:57 UTC, 703895 people within 100km.

Description: On 5/18/2016 7:57:05 AM, an earthquake occurred in Ecuador potentially affecting 703895 people within 100km. The earthquake had Magnitude 6.7M, Depth:32.42km.

Geolocation: 0.4282 -79.765

Details › Delete ›

Title: Orange alert for tropical cyclone ROANU-16. Population affected by Category 1 (120 km/h) wind speeds or higher is 0.

Description: From 18/05/2016 to 21/05/2016, a Tropical Storm (maximum wind speed of 102 km/h) ROANU-16 was active in NorthIndian. The cyclone affects these countries: Myanmar, India, Bangladesh (vulnerability High). Estimated population affected by category 1 (120 km/h) wind speeds or higher is 0.

Geolocation: 23.3 92.5

Details › Delete ›

Disaster Information Service

Threat Landscape

Click on incident to show details

Incident information:

Title:

Description:

Radius (m):

Add incident

Figure 37: Management UI for incident handling

The idea behind this implementation is to provide a dashboard which is only accessible by authorized users to manage incident information. User account information and credentials are stored in the database. Even it is only for internal management purposes, sensitive information, e.g. passwords are stored as hash value + salt to ensure a certain level of security within this prototype. The cryptologic hash function was implemented with an Node.js module which is based on the SHA3 standard [55] with 512 bit length.

This management server was also developed with Node.js framework. In addition to the REST API for browser communication, web-sockets technology (based on NPM module socket.io [56]) is used to provide possibility for instant server updates on incident information. The server is accessible over TCP/IP port 443 and secured with self-signed certificate. The client-side graphical user interface is built on HTML technology and CSS.

4.2. Endpoint/Middleware

As specified in the requirements, the middleware is built in a lightweight and portable way to be able to be integrated on different systems. To build a uniform system, interacting components are combined on one platform. The basic system design of the prototype is illustrated in the following figure:

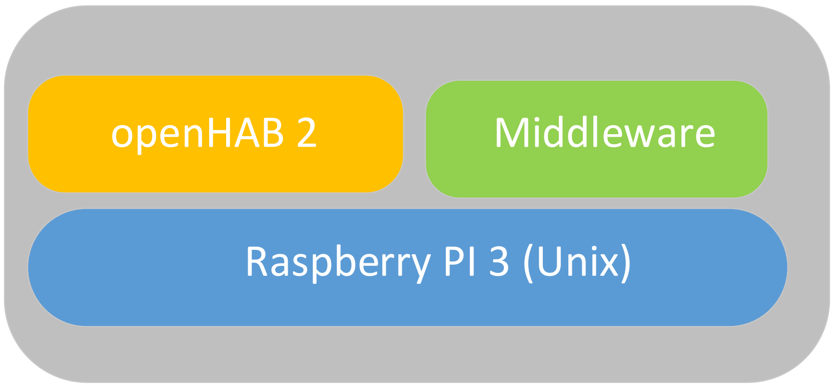


Figure 38: Architecture of the endpoint

Hardware

As it is common in smart home management systems hardware resources are limited. To setup a realistic test environment Raspberry PI Version 3 has been chosen as unified platform. Raspberry is a single-board computer providing the following specifications [57]:

- SoC: Broadcom BCM2837
- CPU: 4× ARM Cortex-A53, 1.2GHz
- GPU: Broadcom VideoCore IV
- RAM: 1GB LPDDR2 (900 MHz)
- Networking: 10/100 Ethernet, 2.4GHz 802.11n wireless
- Bluetooth: Bluetooth 4.1 Classic, Bluetooth Low Energy
- Storage: microSD
- GPIO: 40-pin header, populated
- Ports: HDMI, 3.5mm analogue audio-video jack, 4× USB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)

Smart Home Simulation

In order to simulate smart home functionality, the open-source tool open Home Automation Bus (openHAB) has been used. OpenHab is a stand-alone platform with an integrated API and web-based graphical user interface (GUI) to view the status and manage components (sensors and actuators) from different smart home vendors.

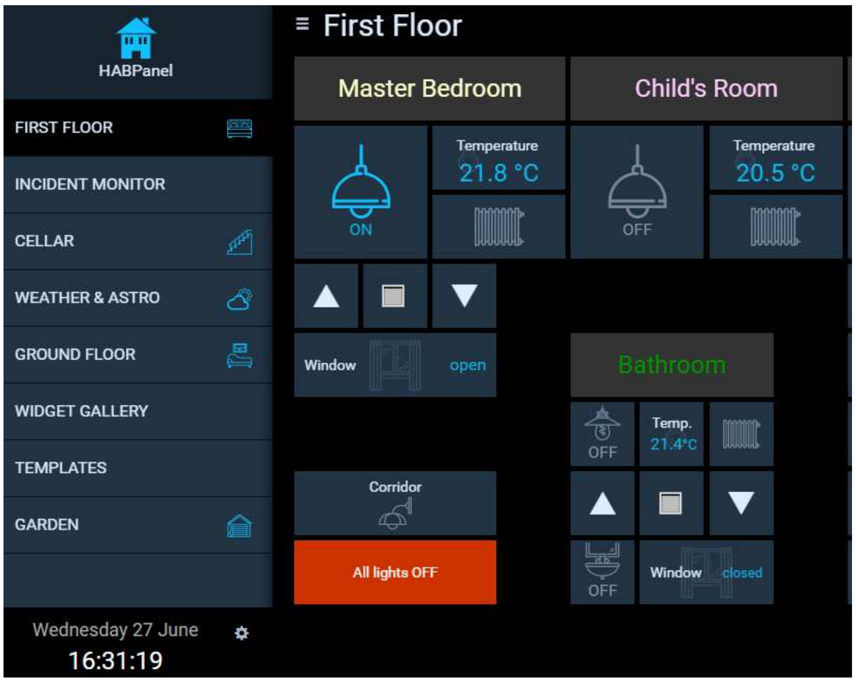


Figure 39: OpenHab Smart Home Simulation

As shown in Figure 39, the interactive dashboard can be build and adapted based on your needs to visualize a smart home. For the simulation with the endpoint/middleware, the default smart home template was used. OpenHab has a built-in API to enable status tracking and configuration of virtual sensors and actuators. The following example gives an idea how the OpenHAB API is used to get a state of a defined sensor or actuator: `http://<ip-address>:<port>/basicui/CMD?<sensor>.` This API is used to perform and demonstrate some risk mitigation tasks by the middleware.

In addition, the middleware is integrated into this web-based dashboard in order to visualized incident information as well. So if a smart home is affected by an incident the related information can be shown in this dashboard.

Communication Middleware

The middleware is built on the Node.js/Express.js framework as well. The reason for that is based on the fact that Node.js/Express can be on different operating systems, e.g. Windows and Unix-based systems. Furthermore, the middleware requires in addition to the incident handling features (e.g. HTTP requests), a sort of visualization and human interaction interface. This is realized via a simple web-based GUI. Of course other development software & tools can be used to implement the endpoint logic. But for this prototype Node.js in combination with openHAB have been used.

The endpoint is required to send requests in order to receive incident information. As it has been specified in section 3.3. , the endpoint send frequent HTTPS requests on a specified interval. In order to prevent concurrency issues on the server-side because of the fixed interval, a delay has been implemented.

```
function startInterval(){

   var counter = 50000;

   var delayFunction = function(){

   //set next runtime to 50sec + a random delay between 1sec and 20sec

   var delay = 50000 + randomNumberGenerator(0,20)*1000;

      //check for update

      run(function(result){

         console.log("Next update in "+delay/1000 +" sec" );

         timeout = setTimeout(delayFunction, delay);

         });

        }

   setTimeout(delayFunction, counter);

}
```

Listing 5: Client request sequence for determining changes

The initial start interval has been set to 50 seconds. In addition, a random number is calculated between 0 and 20 seconds, which is added to the initial value. The result specifies when the next asynchronous request will be sent. This should ensure a slight variance when requests arrives at the server-side in case endpoints are coincidentally urged by the algorithm to send exactly at the same time.
In the following an example of a HTTPS client request for incident meta-info for a specific region is shown:

```
    request('https://'+server+'/incidents/'+region,{headers:{'etag':etag}},function() {
```

Listing 6: HTTP request structure

The `etag` value is specified in the request header which is compared on the server-side to decide if it is required to responds with an HTTP 200 status, including the incident meta-info list or with HTTP status 304 Not Modified.

Distance calculation based on geographical coordinates

In case the endpoint receives a new list of meta-information of related incidents he is interested in, each incident have to be calculated to determine if the endpoint is in the influence area of the incident. This is based on the Haversine formula as already mentioned in section **Fehler! Verweisquelle konnte nicht gefunden werden.**

As required for the distance calculation and furthermore to determine if the endpoint is affected the following parameters are essential: Endpoint latitude and longitude; incident latitude, longitude and radius. As well as the median Earth radius with 6371 kilometers.

The following example is a simplified view of the implementation:

```
function calculation(){

  latDelta = (lat2 - lat1) * Math.PI / 180;

  lonDelta = (lon2 - lon1) * Math.PI / 180;

  lat1Rad = lat1 * Math.PI / 180;

  lat2Rad = lat2 * Math.PI / 180;

  a = Math.sin(latDelta/2)*Math.sin(latDelta/2)+Math.sin(lonDelta/2)*Math.sin(lonDelta/2)*

      Math.cos(lat1Rad) * Math.cos(lat2Rad);

  c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1 - a));

  distance = earthRadius * c;

}
```

Listing 7: Geographical distance calculation

Incident details and visualization

Based on the result of the distance calculation, even detailed information about an incident is required. In this case an additional request have to be sent to the disaster information service including the incident ID.

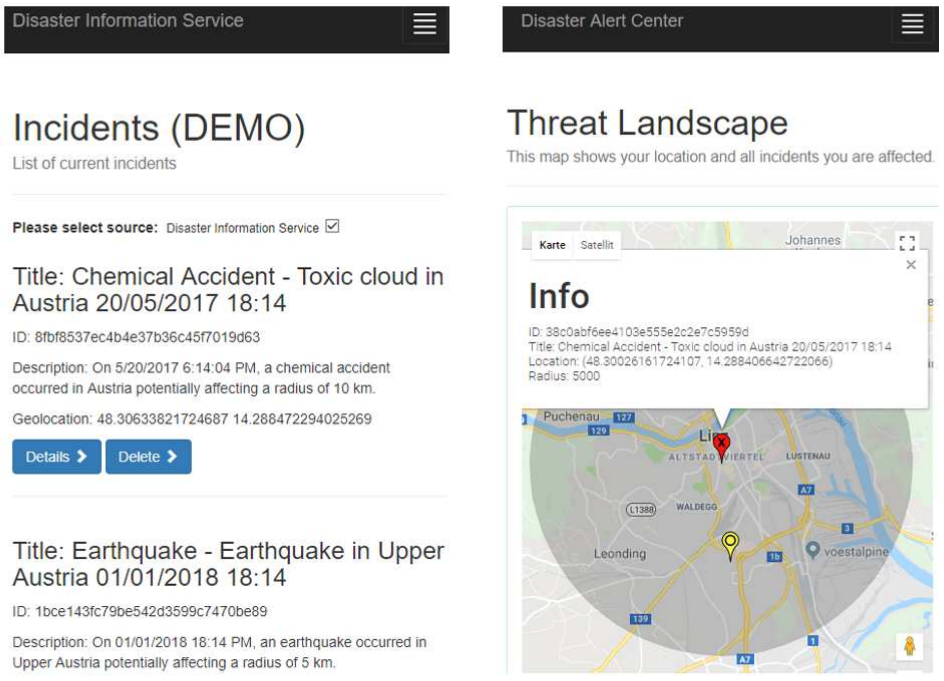
The implementation for the request is illustrated below:

```
request({url: 'https://'+server+'/incident/'+alertItem.id},function(incident) {
```

Listing 8: Request structure for gathering incident details

As identified in the requirements section, the endpoint should provide a possibility for the user to interact with the system in order to specify the location of the endpoint, the incident information of interest or to simply view incident information. So, if the response form the server is successful, the information of the incident is shown in the client-side and web-based UI.

The following figures give an example when an endpoint is affected by an incident:


Disaster Information Service

Incidents (DEMO)

List of current incidents

Please select source: Disaster Information Service ☑

Title: Chemical Accident - Toxic cloud in Austria 20/05/2017 18:14

ID: 8fbf8537ec4b4e37b36c45f7019d63

Description: On 5/20/2017 6:14:04 PM, a chemical accident occurred in Austria potentially affecting a radius of 10 km.

Geolocation: 48.30633821724687 14.288472294025269

Title: Earthquake - Earthquake in Upper Austria 01/01/2018 18:14

ID: 1bce143fc79be542d3599c7470be89

Description: On 01/01/2018 18:14 PM, an earthquake occurred in Upper Austria potentially affecting a radius of 5 km.

Disaster Alert Center

Threat Landscape

This map shows your location and all incidents you are affected.

Info

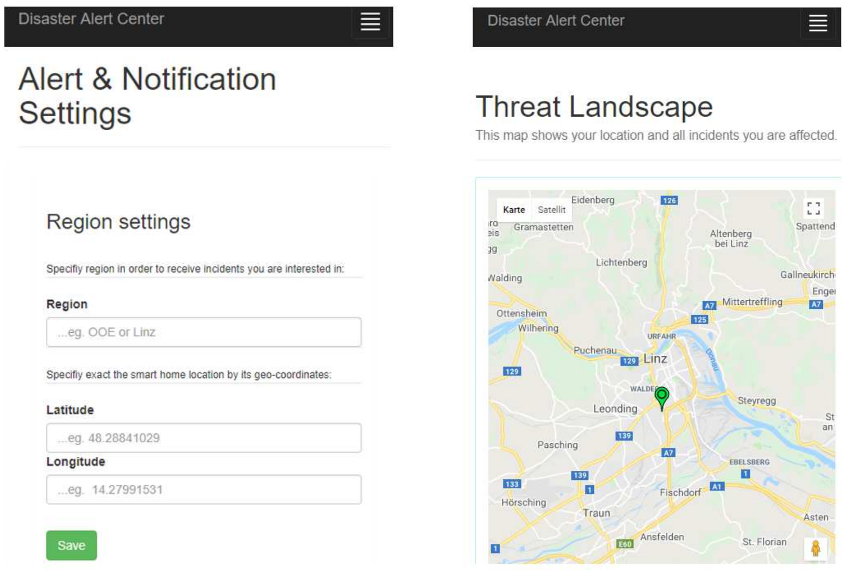
ID: 38c0abf6ee4103e555e2c2e7c5959d
Title: Chemical Accident - Toxic cloud in Austria 20/05/2017 18:14
Location: (48.30026161724107, 14.288406642722066)
Radius: 5000

Figure 40: List of incidents and visualization map

Incident information is shown in a list and within a map. Based on the endpoint settings, only incidents warnings that are of interest and pose a threat are displayed.

Endpoint Location Settings

In order to determine if the endpoint is affected by an incident the location (latitude and longitude) of the endpoint are required. Therefore, web-based configuration page has been developed to provide the owner of the system to set the location of the endpoint and specified the region of interest.


Disaster Alert Center

Alert & Notification Settings

Region settings

Specifiy region in order to receive incidents you are interested in:

Region

...eg. OOE or Linz

Specifiy exact the smart home location by its geo-coordinates:

Latitude

...eg. 48.28841029

Longitude

...eg. 14.27991531

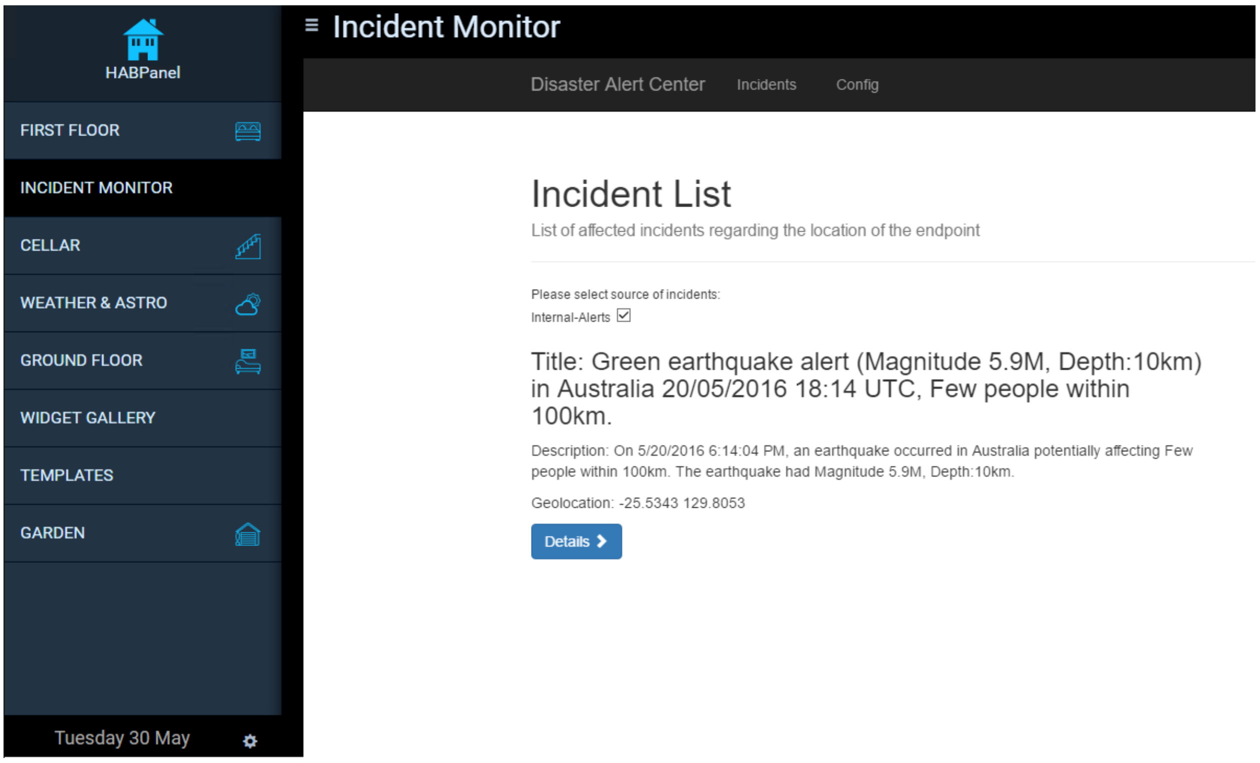
Disaster Alert Center

Threat Landscape

This map shows your location and all incidents you are affected.

Figure 41: Endpoint Configuration

To provide a kind of persistency the values are stored in a JSON file on the local system in this prototype. These configuration is loaded when the endpoint starts.



Incident List

List of affected incidents regarding the location of the endpoint

Please select source of incidents:
Internal-Alerts ☑

Title: Green earthquake alert (Magnitude 5.9M, Depth:10km) in Australia 20/05/2016 18:14 UTC, Few people within 100km.

Description: On 5/20/2016 6:14:04 PM, an earthquake occurred in Australia potentially affecting Few people within 100km. The earthquake had Magnitude 5.9M, Depth:10km.

Geolocation: -25.5343 129.8053

Figure 42: openHAB in combination with the middleware

The benefit of a modular middleware design is that it can be included in different systems, for example in openHAB. The web-based user interface can be simply added in a `Webview` element as illustrated in Figure 42.

4.3. P2P Information Distribution and Sharing

In order to provide peer-to-peer communication capabilities, a proof of concept has been implemented. The disaster information service has been extended with a separated instance (beside the client/server architecture) for seeding and publishing incident information via DHT and the BitTorrent client. This server is integrated in the existing architecture and is connected to the backend database in order to gather incident data. In principle, the peer-to-peer publishing node can coexists with the client/server infrastructure, bound to the same database, but the information distribution works differently.

The implementation is built on node.js JavaScript framework. The proof of concept is based on a torrent client called Webtorrent [58] written in JavaScript. This open source project is published under MIT License terms including the following features:

- Torrent client for node.js & the browser
- Handle multiple torrents simultaneously, efficiently
- Pure JavaScript
- Supports advanced torrent client features (download & seed)

- Magnet URI support
- Peer discovery via DHT and Tracker

WebTorrent’s capabilities are used to create a proof of concept for peer to peer communication in context with the disaster alerting system.

Publishing Regional Contact Information

Because endpoints require an identifier to ask for information changes for its region via the DHT network, the targetID have to be requested first. Therefore, the disaster information service could publish this contact information on its web-service API. The following listing should illustrate a possible information structure and its content.

```
"OOE": {
  "targetID":6b863cd20ed75e5cc0128738c49ceb3a4007579217363592…",
  "city": {
          "linz":{ "targetID": "f1ea5b764f68852a47e0dd80554d…………",
                    "trackers": "udp%3A%2F%2Fexplodie.org%3A6969"
           },
           "wels":{ "targetID": "aa99ea5b764eeeeeeeee0554d3ab………",
                    "trackers": ""
           }
  }
}
```

Listing 9: Initial contact information

So, endpoints first contact the web-service via HTTPS request to get the targetID for the region of interest. This targetID then can be used to determine changes of stored items in the DHT network.

Peer-to-Peer Information Sharing

The disaster information service is responsible for sharing incident meta-information with its clients. Therefore, the following functionality of the peer-to-peer master instance have been implemented:

Methods	Description
`reloadIncidentMetaInfo()`	This method is used to connect to the database and load current incident information.
`seedIncidentInfo()`	Writes incident info to a file, creates a torrent (magnet link, infohash, tracker URLs) based on the file and opens a seeding channel.
`publishNewInfoHash()`	The infohash of the torrent created is published in the DHT network associated with the targetID (hashed public key of the publishing node).

Table 8: P2P publishing methods

The `reloadIncidentMetaInfo()`utilizes the same DBUtil.js mongoDB middleware to connect to the backend database. The `seedIncidentInfo()`method serialize the data on disk in JSON format first. Afterwards, the seeding client creates a torrent magnet link (example can be seen in Listing 10**Fehler! Verweisquelle konnte nicht gefunden werden.**) and waiting for peer connections on TCP port 54444.

```
magnet:?xt=urn:btih:f1ea5b764f68852a47e0dd80554d3ab67c6c4fa6&dn=ooe.json&tr=udp%3
A%2F%2Fexplodie.org%3A6969&tr=wss%3A%2F%2Ftracker.openwebtorrent.com
```

Listing 10: Magnet link sample

This magnet link can be used by other peers to find and download the corresponding information. In this example, the magnet link contains the BitTorrent infohash (*btih),* file name (*dn*) and optional tracker information (*tr*). Now, these contact information have to be transmitted to the endpoints. This is happening in the `publishNewInfoHash()`function.

The publishing mechanism is based on the BitTorrent-DHT module integrated in the WebTorrent bundle. As described in section 2.3.3. the module including features to store and request arbitrary data and in form of mutable items on closest nodes in the DHT network.

Sign and Store mutable items in the DHT network

In order to store mutable items, the payload have to be signed. The node module BitTorrent-DHT module provides a function to sign the infohash using public- and private-key pair. The key-pair should be created once per region and do not change over time. Which means, each incident information about a region (e.g. OOE) is signed by the same key-pair, because of the fact that the public key is a reference point for peers to ask for infohash updates via DHT network and to verify integrity and originator of the infohash. Therefore, the creation of key-pairs and its association with a region is more a static task and have to be done once. For example, region *OOE* belongs to key-pair <*private key X*> and <public key *X*> as described in section 4.3.

An example of the payload of the mutable item containing at least the following attributes:

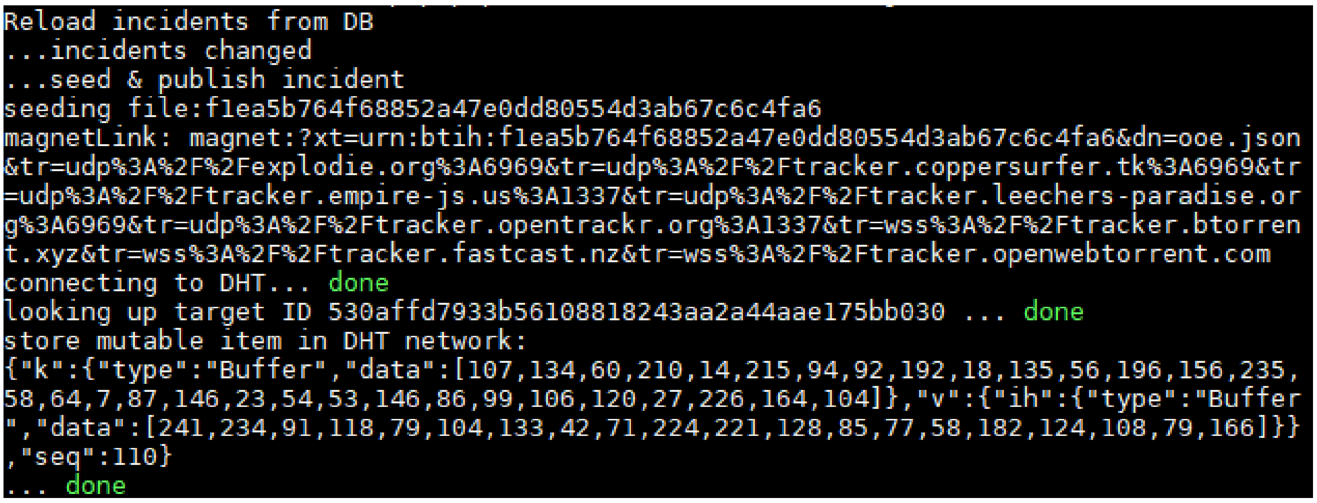
```
{
       k: <publicKey of originator>,
       v: { ih: <current infohash of file, e.g. ooe.json>},
       sign: <signature>,
       seq: <sequence number (increased by +1)>
}
```

Listing 11: DHT mutable item request

The generated infohash of the current (magent link) is wrapped by the DHT mutable item, and will be updated if the content of the incident information in (ooe.json) changes. Signing the content ensures that only the originator can publish new content related to the DHT mutable item. The sequence number defines the version of the item, so it is possible for requesting nodes to identify the latest one. So, mutable items can be updated, without changing their related DHT keys.

These mutable items in the DHT network can be identified and retrieved by other nodes knowing the targetID. The peer needs to ask for the targetID of its region the first time when it will participate in the disaster alerting network.

The following figure should illustrates the publishing sequence:



```
Reload incidents from DB
...incidents changed
...seed & publish incident
seeding file:f1ea5b764f68852a47e0dd80554d3ab67c6c4fa6
magnetLink: magnet:?xt=urn:btih:f1ea5b764f68852a47e0dd80554d3ab67c6c4fa6&dn=ooe.json
&tr=udp%3A%2F%2Fexplodie.org%3A6969&tr=udp%3A%2F%2Ftracker.coppersurfer.tk%3A6969&tr
=udp%3A%2F%2Ftracker.empire-js.us%3A1337&tr=udp%3A%2F%2Ftracker.leechers-paradise.or
g%3A6969&tr=udp%3A%2F%2Ftracker.opentrackr.org%3A1337&tr=wss%3A%2F%2Ftracker.btorren
t.xyz&tr=wss%3A%2F%2Ftracker.fastcast.nz&tr=wss%3A%2F%2Ftracker.openwebtorrent.com
connecting to DHT... done
looking up target ID 530affd7933b56108818243aa2a44aae175bb030 ... done
store mutable item in DHT network:
{"k":{"type":"Buffer","data":[107,134,60,210,14,215,94,92,192,18,135,56,196,156,235,
58,64,7,87,146,23,54,53,146,86,99,106,120,27,226,164,104]},"v":{"ih":{"type":"Buffer
","data":[241,234,91,118,79,104,133,42,71,224,221,128,85,77,58,182,124,108,79,166]}}
,"seq":110}
... done
```

Figure 43: DIS - P2P information publishing (console)

Retrieve and Share Incident Information

At the endpoint site (requesting peer), the same methodical approach can be used to in order to receive and share incident information. In contrast to the disaster information service, an endpoint has to ask the DHT network for updated torrent information and to determine if a new version of incident information for its region has been released. So, beside seeding functionality in order to share incident information with other endpoints, functionality to check and download the torrent are required.

The mentioned functionality can be triggered by following methods:

Methods	**Description**
`checkTorrentInfo()`	This method is used to check if torrent infohash has been updated via DHT network
`downloadIncidentInfo()`	Search for other peers in the swarm for the given infohash, download and write the file to disk.
`seedIncidentInfo()`	Create a torrent of the downloaded file and seed it.

Table 9: Endpoint P2P download and share incidents

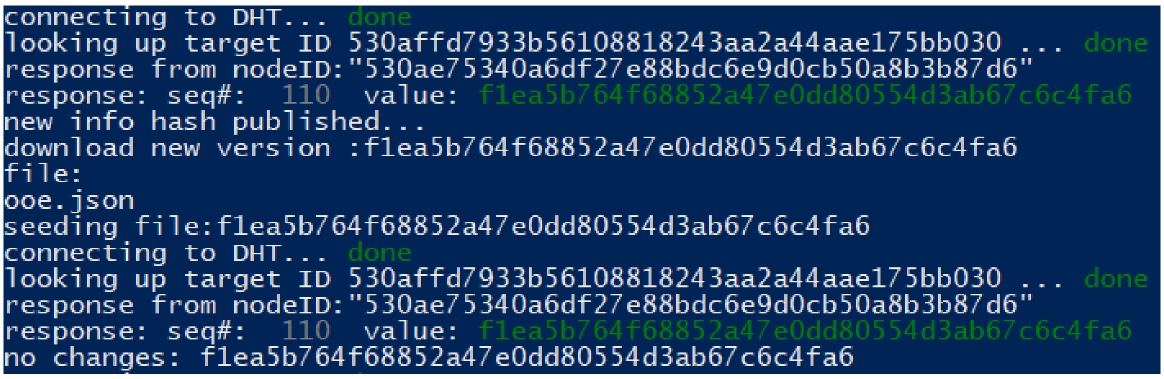
As already mentioned, the targetID has to be pulled from the disaster information service once. The *`checkTorrentInfo()`* connects to the DHT network and asking for mutable items which can be identified by the *targetID*.

The following example represents a typical payload structure of a node response, related to [45]:

```
{
                "id": < id of sending node>,
                "k": <public key>,
                "nodes": <IPv4 nodes close to 'target'>,
                "seq": <monotonically increasing sequence number>,
                "sig": <signature>,
                "v": <ih: infohash>
}
```

Listing 12: Mutable item response structure

The responses from the closest nodes related to the *targetID* are evaluated if it contains a mutable item by checking attributes like `k, sig`, *v.* If these attributes are not available it can be assumed that it is an immutable item or the node responding do not have a DHT item stored. The response also includes a list of closest nodes which are iteratively ask if the responding node cannot successfully offer the mutable item. By verifying the signature, infohash and sequence number it can be ensured that the contained infohash is issued by its originator and is the latest version. Otherwise the response can be seen as invalid.



```
connecting to DHT... done
looking up target ID 530affd7933b56108818243aa2a44aae175bb030 ... done
response from nodeID:"530ae75340a6df27e88bdc6e9d0cb50a8b3b87d6"
response: seq#:  110  value: f1ea5b764f68852a47e0dd80554d3ab67c6c4fa6
new info hash published...
download new version :f1ea5b764f68852a47e0dd80554d3ab67c6c4fa6
file:
ooe.json
seeding file:f1ea5b764f68852a47e0dd80554d3ab67c6c4fa6
connecting to DHT... done
looking up target ID 530affd7933b56108818243aa2a44aae175bb030 ... done
response from nodeID:"530ae75340a6df27e88bdc6e9d0cb50a8b3b87d6"
response: seq#:  110  value: f1ea5b764f68852a47e0dd80554d3ab67c6c4fa6
no changes: f1ea5b764f68852a47e0dd80554d3ab67c6c4fa6
```

Figure 44: Endpoint P2P communication

If everything is valid, the endpoint can go on and trigger the `downloadIncidentInfo()`method. Based on the infohash received from the last lookup, a DHT discovery sequence is initiated to find related peers and download the desired information. As soon as the information has been downloaded, the endpoint start sharing the content by triggering the `seedIncidentInfo()` method.

5. Evaluation

In order to show possibilities and limitations of the centralized disaster information service, an evaluation of performance metrics was made. In the following, an overview of valuable performance metrics, the test environment, test scenario and related results are presented. In addition, observations of the peer-to-peer information sharing approach are highlighted in this section.

The disaster information service plays a central role in this hybrid system design. Although, peer-to-peer communication capabilities are discussed and available, client-server communication is required in order to provide at least incident meta-information to its endpoints. Depending on the number of the participating endpoint, therefore, the number of requests to the system will increase. In order to be able to estimate how the system behaves when it comes to peak loads, usually performance and load tests are a good option.

Performance indicators

Simulated load test try to give an answer as to whether a web service is capable of handling an increasing number of requests and at what level it comes to loss of performance. The performance and stability of a system can usually be measured and interpreted using various key performance indicators. Because the disaster information service is realized via web-service, response time and volume metrics are decisive.

The following performance metrics can provide information about how a system behaves during a load test [59]:

- Response time: The duration how long it takes to get an answer from the server
- Timeout errors: Number of requests timed out
- Error rate: Percentage of total number of request failed
- Requests per second: Number of requests per second sent to the server
- CPU Utilization: Percentage of CPU usage during the test

Test environment

The goal is to measure the disaster information service performance during a high load of requests sent by the endpoints. To obtain meaningful values, a test environment have to be set up. The virtual test environment requires two essential components. A work-load generator simulating endpoints sending requests and a virtual machine hosting the disaster information service. The environment was created on the Microsoft Azure cloud to build a realistic situation. The platform provides several features even including load testing capabilities. Loader.io [60], is work-load generator which can be integrated to enable performance test within the Azure platform. In its free plan, it supports testing scenarios generating work-loads up to 10k concurrent request per second. The virtual host running the disaster information service has 6 virtual CPUs, 16 GByte memory and 1 GBit network connection, which corresponds to a mid-size configuration.

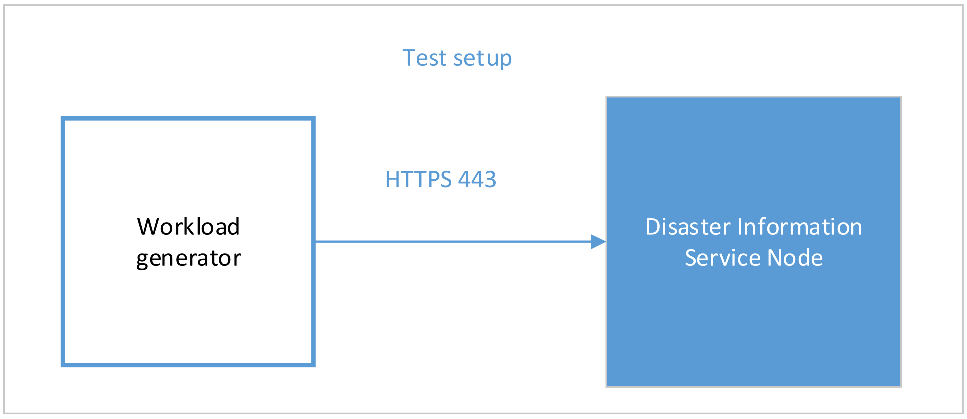


Figure 45: Performance test setup

Figure 45 illustrates the test setup and the communication flow. The work-load generator will send HTTPS request via network to its target and measure related response metrics as mentioned above. On the other hand, the CPU utilization of the service host has been monitored during a test run.

Test configuration and results

The central questions are: What impact does an increasing load have on the service performance and at which request level you have to expect performance loss or even worse service outages. The derived information of such tests should then be considered for building scalable service infrastructure. In order to show the utilization and performance at increasing load, different levels have been configured (100, 1000 and 10k requests per seconds). The duration of a test run was set to 1 minute. This configuration allows a simple extrapolation of the results at the end. HTTPS GET request have been sent to *https://<server>/incidents/ooe/<id>* to simulate requests for incident information at a constant load.

The results of the test runs are presented in the following table:

Test Case #	Requests per second sent #	Avg. response time (ms)	Timeouts (ms)	Error rate %	CPU utilization %
1	100	85	0	0	~ 5 %
2	1000	87	0	0	~ 25 %
3	10k	1286	5744	4,10%	~ 90 %

Table 10: Test results

As can be seen in Table 10, the values for test run number 1 and 2 do show expected good results. The average response time differs only by 2 milliseconds by an increased request rate of factor 10. All transactions were successful generating a maximum CPU load lower than 5% in the first case and 25% in the second one. At a level of 10k request per second performance indicators show different results. The average response time goes up to ~1.2 seconds, which is actually an acceptable value. The result in test case number 3 also shows that 5744 out of 600k requests sent have been timed out. The error rate goes up to 4.10%, reflecting that request were not answered accordingly or in time. The peak of the CPU utilization during the 1 minute test was at ~90%. This indicates that CPU resources actually were available and implies that possibly other factors, e.g. Network latency or security mechanisms could be sources of interference. However, a maximum of 10k request per seconds is an acceptable result.

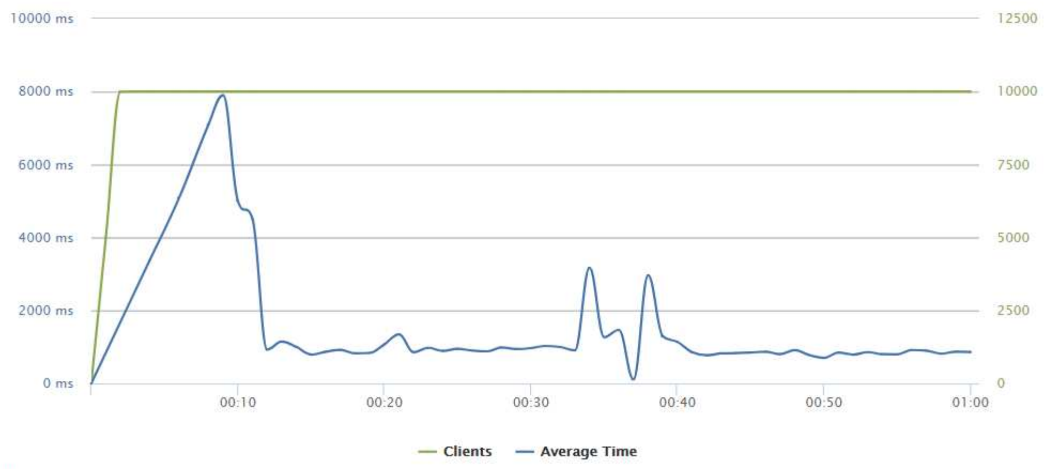


Figure 46: Test case - 10k requests per second

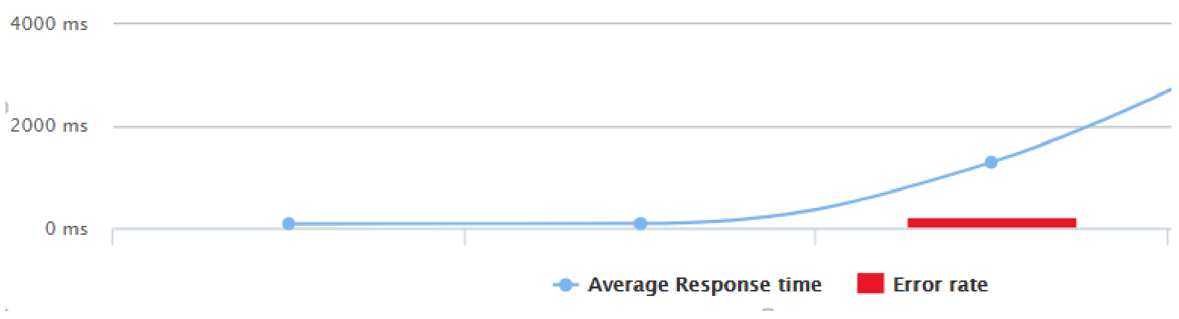


Figure 47: Avg. response time and error rate

Base on the results of the test, a deviation can be made to put the values in the right context. As it is specified in the design for native client-server communication, endpoints are only be allowed to send a request once per minute. Theoretically, using this hosting configuration and setting would allow one single instance supporting up to 600k endpoints in a typical client server communication approach. With an increasing number of participants new instances could be added to the system in order to scale horizontally and server the next 600k clients.

In the age of virtualization and cloud computing, changes to systems are no longer challenging. That means system can dynamically scale in a vertically way, by increasing necessary system resources or in a horizontally way, simply higher the number of serving instances.

P2P information publishing and observations

Despite various methods and mechanisms to make a central system scalable, the fact remains with pure client/server communication that more effort and resources must be made available as the load increases. To reduce this general burden, a decentralized peer-to-peer communication approach is desired to spread information on several contact points.

In the following, observations of the implemented peer-to-peer information sharing approach between the disaster information service and endpoints are described. The evaluation relates to general performance indicators [61, p. 2], such as overall information lookup time, data persistence and content integrity of published and retrieved information via the DHT network.

Lookup performance: The logarithmic routing mechanism of the Kademlia DHT allows to reduce the XOR distance to the target key by ½ per lookup [62, p. 2]. These operations are iterative performing sequential queries of intermediate nodes until no closer contacts can be obtained. In a simple 1:1 test scenario, where the publisher (disaster information service) and a consumer (endpoint) reside on different sites (communicating via internet), the information lookup operation in point of view of a consumer takes up to 5 seconds on average until the desired information (e.g. infohash) can be retrieved. This time span seems an acceptable value in order to determine if some incident information updates are available at the end.

Availability and persistence: As given by design, a put operation is used to store torrent information in the DHT network. The information is typically replicated to the closest nodes of the key space [62, p. 2], in a viewpoint of the disaster information service performing the store operation. During the test, the initial publisher has been turned off, in order proof if data is still available and can be retrieved by an endpoint. The test was successful and it could be approved that even the originator of the information goes offline, data can be retrieved by other closest nodes from the DHT network. This is the result of migrating items from node to node in case of a node becomes unavailable. But in this particular case it has been recognized that stored items are discarded by nodes approximately after an hour and the lookup operation runs into errors or respond with a wrong item (not the latest). Therefore, data items carry meta-information of incident have to be refreshed (published) to stay alive in the network even information have not changed in the back.

Integrity: The concept for decentralized information publishing and consumption via DHT networks intend to use mutable items. As incident information have to be frequently updated and stored different nodes, integrity and authenticity of these object have to be checked. Before nodes store mutable items in its routing table, they use the public key for verifying the originator and content. So, the verification of an item is done at store operations. Which means that it is not possible for endpoints to retrieve modified content performing a *get* request. During the evaluation of these concept it has been noticed that the retrieved information always reflects the content published from its originator. Furthermore, the version of the mutable item containing incident information is verified by the publishing sequence number.

The different information exchange approaches have been evaluated. Both have its advantages and challenges. Whereas the centralized communication approach makes it easy to maintain and publish incident information (versioning), guarantee data integrity and security, the peer-to-peer approach has its strength in availability and scalability, even in a large scenario.

6. Future Work

The concept developed for a disaster alerting system employing home automation systems was implemented in form of a prototype in order to illustrate new possibilities for alerting and protection measures. Various aspects of the design and communication infrastructure were considered, implemented and evaluated. However, it should be noted that this prototype is not a complete product intended for productive use. Although, the techniques used can be considered and implemented for a productive system.

Security mechanisms like TLS encryption has been considered and implemented in the disaster information service. But it would be recommended to evaluate the system especially under the security aspect. Especially in the field of peer-to-peer communication, considerations and improvements regarding security must be deepened. Although the information should in principle be publicly available, the integrity of data during the transmission within the DHT network could be further improved with the use of cryptographic algorithms. For, example a Public Key Infrastructure (PKI) could be implemented to ensure a secure transmission in end-to-end communication.

Furthermore, in order to efficiently integrate smart home automation systems, a technical specification based on the categories of incidents, their recommended self-protection measures and tasks to be performed by the smart home is desirable. This could result in a sort of reference table, to match self-protection tasks with recommended sensor and actuator activity.

7. Conclusion

The advancing digitization in the private sector and the growing number of home automation systems open up new possibilities in the context of an early warning system. The aim of this thesis was to develop a concept for integrating smart home systems into a disaster alerting process. To achieve this goal, first known general requirements for a disaster alerting system based on literature and national civil protection guidelines were examined. It has become clear that aspects such as awareness (something has happened), informational content (what has happened) and self-protection measures (what can i do to protect myself) have to be essential components within an early warning system. These aspects were examined in context with the current alerting techniques. From this it can be seen that some of these aspects are more or less supported but from the point of view of the persons involved it is sometimes difficult to interpret them in order to be able to derive suitable self-protection measures, without the addition of further sources of information.

New alerting approaches such as via SMS or smartphone apps can handle these requirements very well. But, even in this approaches, it is still the duty of the victim to carry out the risk mitigation tasks by himself. With the inclusion of smart home systems, it is also possible to automate these manual activities. Even if use cases for performing automatic risk mitigation tasks are limited, the integration of smart home systems can help to notify (for example, playing alarm sound or flashing lights in case of an alert), visualize information and give an overview about the status of possible critical elements, such as open windows or doors in case of a spreading toxic cloud threatening system owners region.

Beside functional requirements, technical requirements have been derived which should be consider to build a comprehensive system. An additional question of this thesis also refers to which communications infrastructure and systems are required for a scalable and reliable system.

Various communications models were examined to what extent they are suitable for a scalable distributed information system in the context of an early warning system. Opportunities and challenges are have been highlighted. A central challenge in connection with pull-based communication approaches is to provide incident information in a timely and valid manner. Theoretically, different approaches can address this challenges, as has been illustrated for example on a DNS based information distribution approach. But in practice there are always compromises to make. Centralized communication systems, have their strengths in terms of security, trust, manageability and connectivity and do not facing information versioning problems. However, a central point of contact can lead to performance issues with an increasing number of requests. Hence, additional effort is required to ensure the availability and scalability of a centralized system, which can be achieved using the mentioned distribution and balancing techniques. In contrast, decentralized information systems realized for example via peer-to-peer based communications models provide maximum scalability, because each participating peer can act as a potential information distributor within the network. But here too, peer-to-peer typical problems such as peer discovery, end-to-end connections and especially the distribution of the right information must be taken into account. Therefore, a prototype has been developed that combines both client / server approaches and peer-to-peer capabilities. In addition to the reliable transfer of data, the correct interpretation is crucial. The division between meta-information and incident details allows an efficient determination of whether certain endpoints are affected by incidents or not. In addition to the reliable transfer of data, the correct interpretation is crucial. The division between meta-information and incident details allows an efficient determination of whether certain endpoints are affected by incidents or not, without sending unnecessary data over the network, which may not be relevant.

This results in a hybrid system that can provide incident information through various communication interfaces. In particular, decentralization to provide information updates is a huge advantage in the case of a disaster alerting system. This means that the load can be distributed evenly among a large number of participants. However, as the results of the evaluation of the central information service have shown, a large amount of request (without peer-to-peer communication) can be traded. Various strategies and techniques have been developed for this purpose. Thus, a vertical or horizontal scaling of systems can be achieved at the present time, where resources can be automatically added, without much effort.

In summary, it can be said that digitization and automation have meanwhile taken hold in all areas, which opens up new possibilities in various contexts. Traditional well-established alerting approaches can be improved and expanded as a result of this progress.

References

[1] United Nations International Strategy for Disaster Reduction, “Global Survey of Early Warning Systems,” *United Nations*, no. https://www.zivilschutz-ooe.at/wp-content/uploads/2017/10/AKW-Unfall-1.pdf, p. 56, 2006.

[2] T. E. Drabek and G. J. Hoetmer, *Emergency management: Principles and practice for local government*. International city management association Washington, DC, 1991.

[3] Bundesministerium des Innern, “Leitfaden Krisenkommunikation,” 2018. [Online]. Available: https://www.bmi.bund.de/SharedDocs/downloads/DE/publikationen/themen/bevoelkerungsschutz/leitfaden-krisenkommunikation.html. [Accessed: 03-Feb-2017].

[4] S. Jachs, “KOORDINATION VON KRISEN- UND KATASTROPHENSCHUTZ- MANAGEMENT,” *Bundesministerium für Inneres*, 2011. [Online]. Available: https://www.bmi.gv.at/204/Internationales_Katastrophenmanagement/start.aspx. [Accessed: 03-May-2017].

[5] BMI, “Bundesministerium für Inneres - Zivilschutz Austria,” 2017. [Online]. Available: http://www.bmi.gv.at/cms/bmi_zivilschutz/. [Accessed: 04-Feb-2017].

[6] V. Grasso, A. Singh, and J. Pathak, “Early Warning Systems A State of the Art Analysis and Future Directions,” 2012.

[7] GDACS, “Global Disaster Alert and Coordination System,” 2016. [Online]. Available: http://portal.gdacs.org/about. [Accessed: 05-Feb-2017].

[8] OASIS - Advancing open standards for the information society, “Common Alerting Protocol,” 2016. [Online]. Available: http://docs.oasis-open.org/emergency/cap/v1.2/CAP-v1.2-os.html. [Accessed: 20-Feb-2017].

[9] EUMETNET, “Meteoalarm - Alerting europe for extream weather,” *The Network of European Meteorological Services*, 2016. [Online]. Available: http://www.meteoalarm.eu/?lang=de_GE. [Accessed: 15-Feb-2017].

[10] FEMA, “Integrated Public Alert and Warning System,” *Federal Emergency Management Agency*, 2015. [Online]. Available: https://www.fema.gov/integrated-public-alert-warning-system. [Accessed: 02-Mar-2017].

[11] KATWARN, “KATWARN - Katastrophen Warn App,” 2016. [Online]. Available: http://www.katwarn.de. [Accessed: 05-Mar-2017].

[12] BIWAPP, “BIWAPP - Bürger Info und Warn App,” 2016. [Online]. Available: http://www.biwapp.de. [Accessed: 23-Mar-2017].

[13] Fraunhofer Institute, “EWF - Emergency Warning Functionality,” 2016. [Online]. Available: http://www.audioblog.iis.fraunhofer.de/nab-show-2014-ewf/. [Accessed: 10-Apr-2017].

[14] C.-Y. Lin, E. T.-H. Chu, L.-W. Ku, and J. W. S. Liu, “Active disaster response system for a smart building,” *Sensors*, vol. 14, no. 9, pp. 17451–17470, 2014.

[15] B. Aschendorf, “Übersicht über Gebäudeautomationssysteme,” in *Energiemanagement durch*

Gebäudeautomation, Springer, 2014, pp. 197–779.

[16] Techopedia, "Home Automation System - Definition," 2016. [Online]. Available: https://www.techopedia.com/definition/29999/home-automation-system. [Accessed: 23-May-2017].

[17] Bussysteme, "Aktoren und Sensoren," *Bussysteme*, 2016. [Online]. Available: https://www.smarthome-geraete.de/bussysteme/. [Accessed: 23-May-2017].

[18] Wikipedia, "Smart Homes," 2016. [Online]. Available: https://de.wikipedia.org/wiki/Smart_Home. [Accessed: 04-Jun-2017].

[19] H. Brandstetter, "Heimautomationssysteme - Theorie und Praxis," pp. 1–153, 2008.

[20] Iridiummobile, "Setting up Connection to KNX," 2016. [Online]. Available: http://wiki2.iridiummobile.net/Setting_up_Connection_to_KNX. [Accessed: 04-Jun-2017].

[21] OpenHAB, "OpenHAB - Introduction." [Online]. Available: http://www.openhab.org. [Accessed: 19-Jun-2017].

[22] J. P. Martin-Flatin, "Push vs. pull in Web-based network management," *Integr. Netw. Manag. VI. Distrib. Manag. Networked Millenn. Proc. Sixth IFIP/IEEE Int. Symp. Integr. Netw. Manag. (Cat. No.99EX302)*, no. May, pp. 3–18, 1999.

[23] J. DeNero, "Distributed and Parallel Computing," *Berkeley.edu*, 2016. [Online]. Available: http://wla.berkeley.edu/~cs61a/fa11/lectures/communication.html#distributed-computing. [Accessed: 13-Aug-2017].

[24] A. Schill and T. Springer, *Verteilte Systeme: Grundlagen und Basistechnologien*. Springer-Verlag Berlin Heidelberg, 2007.

[25] S. Von Brauk and C. Neudert, "Prinzipien-skalierbarer-Architektur," *jaxenter.de*, 2014. [Online]. Available: https://jaxenter.de/prinzipien-skalierbarer-architektur-848. [Accessed: 20-Aug-2017].

[26] Codeproject, "Multi-Tier Architectures," 2016. [Online]. Available: http://www.codeproject.com/Articles/430014/N-Tier-Architecture-and-Tips. [Accessed: 20-Aug-2017].

[27] P. Mandl, *Masterkurs Verteilte betriebliche Informationssysteme - Prinzipien, Architekturen und Technologien*. Vieweg+Teubner Verlag | Springer Fachmedien Wiesbaden GmbH, Wiesbaden, 2009.

[28] Diffen, "Comparison TCP vs. UDP," 2016. [Online]. Available: http://www.diffen.com/difference/TCP_vs_UDP. [Accessed: 10-Oct-2017].

[29] Wikipedia, "Transport Layer Security," 2017. [Online]. Available: https://de.wikipedia.org/wiki/Transport_Layer_Security. [Accessed: 10-Oct-2017].

[30] Selfhtml, "Grundlagen des Domain Name System," 2017. [Online]. Available: https://wiki.selfhtml.org/wiki/Grundlagen/Domain_Name_System. [Accessed: 15-Oct-2017].

[31] Microsoft Technet, "Recursive and Iterative Queries," 2010. [Online]. Available: https://technet.microsoft.com/de-de/de/library/cc961401.aspx. [Accessed: 18-Oct-2017].

[32] RFC-1034, "Domain Names - Concepts and Facilities," 1987. [Online]. Available:

https://tools.ietf.org/html/rfc1034. [Accessed: 27-Oct-2018].

[33] Wikipedia, “Anycast Routing,” 2017. [Online]. Available: https://de.wikipedia.org/wiki/Anycast. [Accessed: 27-Oct-2017].

[34] A. Bhatia, “Different CDN technologies: DNS Vs Anycast Routing,” 2017. [Online]. Available: https://anuragbhatia.com/2014/03/networking/different-cdn-technologies-dns-vs-anycast-routing/. [Accessed: 03-Nov-2017].

[35] Statistics Austria, “AUSTRIA Data. Figures. Facts,” 2018. [Online]. Available: https://eu2018.statistik.at/fileadmin/euratspraesidentschaft/downloads/austria._data._figures._facts.pdf. [Accessed: 06-Jan-2018].

[36] BitTorrent, “Deutsche BitTorrent FAQ,” 2018. [Online]. Available: http://www.bittorrent-faq.de/#ss1.1. [Accessed: 12-Mar-2018].

[37] B. Cohen, “The BitTorrent Protocol Specification,” *10.01.2008*. [Online]. Available: http://www.bittorrent.org/beps/bep_0003.html. [Accessed: 12-Mar-2018].

[38] A. Loewenstern and A. Norberg, “DHT Protocol,” *BitTorrent.org*, 2008. [Online]. Available: http://www.bittorrent.org/beps/bep_0005.html. [Accessed: 12-Mar-2018].

[39] P. Maymounkov and D. Mazieres, “Kademlia: A Peer-to-Peer Information System Based on the XOR Metric,” p. 13.

[40] J. Fuchs, “A Torrent Recommender based on DHT Crawling,” ETH Zürich, 2015.

[41] Z. Hu, “NAT Traversal Techniques and Peer-to-Peer Applications,” *Telecommun. Softw. Multimed. Lab. Helsinki Univ. Technol.*, 2005.

[42] G. Hazel and A. Norberg, “Extension for Peers to Send Metadata Files,” 2012. [Online]. Available: http://bittorrent.org/beps/bep_0009.html. [Accessed: 15-Mar-2018].

[43] L. Matteis, “Updating Torrents Via DHT Mutable Items,” 2018. [Online]. Available: http://bittorrent.org/beps/bep_0046.html. [Accessed: 13-Mar-2018].

[44] A. Norberg, “BitTorrent extension for DHT RSS feeds,” 2018. [Online]. Available: https://libtorrent.org/dht_rss.html. [Accessed: 12-Mar-2018].

[45] A. Nordberg and S. Siloti, “Storing arbitrary data in the DHT,” 20118. [Online]. Available: http://bittorrent.org/beps/bep_0044.html. [Accessed: 15-Mar-2017].

[46] R. T. Fielding, “Architectural Styles and the Design of Network-based Software Architectures,” *Building*, vol. 54, p. 162, 2000.

[47] Heroku, “Increasing Application Performance with HTTP Cache Headers,” 2017. [Online]. Available: https://devcenter.heroku.com/articles/increasing-application-performance-with-http-cache-headers. [Accessed: 15-Dec-2017].

[48] T. Bray, “The JavaScript Object Notation (JSON) Data Interchange Format,” *Internet Engineering Task Force (IETF)*, 2017. [Online]. Available: https://tools.ietf.org/pdf/rfc8259.pdf. [Accessed: 17-Dec-2017].

[49] N. Chopde and M. Nichat, “Landmark Based Shortest Path Detection by Using A* and Haversine Formula,” *GH Raisoni Coll. Eng. …*, vol. 1, no. 2, pp. 298–302, 2013.

[50] MongoDB, “MongoDB - Datenverwaltung neu erfunden,” 2017. [Online]. Available: https://www.mongodb.com/de. [Accessed: 15-Jan-2018].

[51] Node.js, “About | Node.js,” *Nodejs.org*, 2016. [Online]. Available: https://nodejs.org/en/about/. [Accessed: 25-Jan-2018].

[52] Node Package Manager, “What is npm?,” 2018. [Online]. Available: https://docs.npmjs.com/getting-started/what-is-npm. [Accessed: 25-Jan-2018].

[53] NGINX, “Welcome to the NGINX and NGINX Plus documentation,” 2018. [Online]. Available: https://docs.nginx.com/nginx/. [Accessed: 25-Jan-2018].

[54] R. Nelson, “SSL/TLS Offloading, Encryption, and Certificates with NGINX and NGINX Plus.” [Online]. Available: https://www.nginx.com/blog/nginx-ssl/.

[55] M. J. Dworkin, “SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions,” *Draft FIPS PUB 202*, no. August, 2015.

[56] Socket.io, “Socket.io - How to use,” 2018. [Online]. Available: https://socket.io. [Accessed: 27-Jan-2018].

[57] RaspberryPi, “Raspberry PI 3 Specification Benchmarks,” 2018. [Online]. Available: https://www.raspberrypi.org/magpi/raspberry-pi-3-specs-benchmarks/. [Accessed: 14-Mar-2018].

[58] “WebTorrent - The streaming torrent client. For node.js and the web.” [Online]. Available: https://www.npmjs.com/package/webtorrent. [Accessed: 14-Mar-2018].

[59] “What do Load Testing Metrics tell us about Performance?,” 2018. [Online]. Available: http://loadstorm.com/load-testing-metrics/. [Accessed: 14-May-2018].

[60] Loader.io, “The Loader.io APIhttp://www.news.cs.nyu.edu/~jinyang/pub/iptps04.pdf,” 2018. [Online]. Available: http://docs.loader.io/api/intro.html. [Accessed: 13-May-2018].

[61] J. Pouwelse, P. Garbacki, D. Epema, and H. Sips, “The BitTorrent P2P file-sharing system: Measurements and analysis,” *4th Int. Work. Peer-to-Peer Syst.*, pp. 205–216, 2005.

[62] J. Falkner, M. Piatek, J. P. John, A. Krishnamurthy, and T. Anderson, “Profiling a million user dht,” *Proc. 7th ACM SIGCOMM Conf. Internet Meas. - IMC ’07*, p. 129, 2007.

Curriculum Vitae

Stefan Reinthaler, BSc.
12. Mai 1987
Nationality: Austria
stefan@reinthalernet.at

Current Employment	
10/2015 – now	**Senior Associate, Risk Consulting** KPMG Alpen-Treuhand GmbH
Previous Employments	
03/2014 – 09/2015	**Part-Time Employee in a Datacenter** TTG Tourismus Technologie GmbH, 4040 Linz 1st & 2nd level-support, user and system administration, asset management
02/2007 – 09/2011	**IT System Engineer** AKD Baunetzwerk GmbH, 4040 Linz Account Manager, 1st & 2nd Level Support, Network management (WAN, LAN, VPN), Exchange administration, Client/Server Lifecycle-Management
Education	
03/2014 – now	**Master‘s program Computer Science** Johannes Kepler University Linz, 4040 Linz
03/2014 – now	**Master‘s program Business Informatics** Johannes Kepler University Linz, 4040 Linz
09/2011 – 03/2014	**Bachelor's degree Business Informatics** Johannes Kepler University Linz, 4040 Linz
06/2009 – 09/2011	**General qualification for university entrance** Johannes Kepler University Linz, 4040 Linz
Skills and Interests	
Certificates	ITIL v3 Foundation, Microsoft Certified Professional - Licensing
Languages	German (native), English (fluent), Spanish (A1/A2)
Hobbies	Climbing, Running, Traveling

STATUTORY DECLARATION

I hereby declare under oath that the submitted Master's Thesis has been written solely by me without any third-party assistance, information other than provided sources or aids have not been used and those used have been fully documented. Sources for literal, paraphrased and cited quotes have been accurately credited.

The submitted document here present is identical to the electronically submitted text document.

Place, Date

Signature