Abstract. This paper deals with the development of a decentralized paging transmitter network called "DAPNET" used by the amateur radio service in Germany. The aim is to implement a software, as successor of the already existing system, which will be able to build up a paging service using the current infrastructure (for example the transmitters and receivers). A main requirement is to achieve a decentralized system, which means that the software runs on geographically distributed servers. Accruing data (for example entries from the user administration, defined transmitter groups, call-signs, etc.) have to be synchronized.

After analysing the current infrastructure and collecting requirements, a detailed design process has been performed. As a result the network structure and the software architecture including instructions about how to realize each component were defined. The last step is the concrete implementation of the "DAPNET Core" as well as a first test.

Keywords
Paging, Decentralized Network, Cluster, JGroups, RESTful Webservice, Amateur Radio, POCSAG

1. Introduction
The term "paging" describes the unidirectional transmission of coded signals or messages to a mobile receiver called "pager" or "beeper". The messages are transmitted by "paging systems" consisting of one or more paging transmitters.

Paging systems were widely used until the end of the nineties, before being mostly replaced by today's mobile radio systems like GSM, UMTS and LTE. On first sight, they seem to be clearly superior in all aspects (bidirectional communication, higher data rates, ...). However, paging allows for example the use of passive receivers, the easy set-up of an independent wide-area radio networks by using high power transmitters and gives the possibility to notify many subscribers simultaneously as well as reliably.

These characteristics predestine this technology to be used to alert large groups of people, as required for emergency services, where it is still used today. It is conceivable that there might be new use cases, for example in context of Smart Grids, where paging could be used to simultaneously inform lots of devices about the network state.

German radio amateurs early realised the potential of this technology and take over hardware and frequencies from abandoning commercial operators. After some years of operating a nearly German-wide network more and more transmitters were shut down due to the old incompatible hardware, new regularities concerning the energy consumption of transmitters as well as faulty and unstable software controlling the system.

The aim of this work is to develop a new concept for the paging system and especially to implement the software required to operate a completely decentralized transmitter network. The new paging system is named "DAPNET" ("Decentralized Amateur Paging NETwork").

2. Infrastructure
Initial point of this work has been an analysis of the used technologies and the current infrastructure, which should mostly be supported by the new system. A short overview is given in the following. For detailed information please refer to [1].

2.1. Paging Transmitter Networks
A wide-area paging service can only be realised by a network of several transmitters. This leads to the necessity of a system that accepts requests for messages and distributes them to the transmitters. Such a system is called "paging (transmitter) network" and is the main subject of this paper.

German radio amateurs used to date the so called "Funkrufmaster" (further details see [2]), a software running on several servers, operated by local radio amateur groups. The software offers rudimentary interfaces to accept messages, which are distributed by manually configured for-
wards to other instances of the Funkrufmaster. Each instance again forwards the messages to all transmitters it is controlling.

An orientation at most of the existing commercial paging networks is not possible due to the fact that they are typically realised as single-frequency networks, which means that all transmitters are sending the signal simultaneously on one frequency. The required high-precision synchronisation is achieved by using satellites. The additionally needed receiver hardware at the transmitter stations is too expensive for the radio amateur service. As alternative a time division multiple access scheme (TDMA) is used, which ensures that transmitters with overlapping coverage are not sending at the same time.

For the communication within the paging network the so called "HAMNET" (successor of the old packet radio network) is available, which provides an German-wide IP-network based on radio links.

2.2. Transmitter

For sending messages to the pagers mainly the ERICSSON COMPACT 9000 transmitter has been used in the radio amateur paging network. Since it became obsolescent and has especially a high energy consumption a modern alternative called "RasPager" (see figure 1) was developed, which is also available in a high transmitting power version called "RasPager Digi" [3].

The protocol for controlling the RasPager (called "RasPager protocol") was designed fully compatible to the one of the Ericsson transmitter. It specifies among other things the time synchronisation and the assignment of time slots of the TDMA scheme. A detailed description is given in [3]. From outside perspective the only difference between both devices is that the RasPager can be contacted directly via TCP, while the Ericsson transmitter needs a converter to its serial interface.

The so called "POCSAG protocol" (for further details see [3][4]) defines the coding of messages send over air to the receiver. An analysis of the protocol is needed for a correct estimation regarding the maximum number of messages, which can been transmitted within a single time slot (for details see [1]).

2.3. Receiver

For receiving the POCSAG-coded messages a compatible receiver is necessary. Lots of different models are available on the market. Radio amateurs use mostly the "Skyper" (see figure 2), which was formerly used for a paging service of same name. In addition to the receiving of simple messages the current system time and date are displayed. Furthermore, the user can subscribe for different rubrics, which can be compared to news channels, which inform the user for example about the canteen menu. This comfort feature as well as the easy modification for using the Skyper at the used radio frequency of 439,9875 MHz explains the high popularity.

The main problem has been the lack of documentation of how to code the POCSAG messages to control the many functions. By analysing the source code of the Funkrufmaster and by lots of try and error the full protocol (called "Skyper protocol") has been reconstructed (see [1]).

3. Requirements

As a result of analysing the existing infrastructure, as well as by intercommunication with the radio amateurs, a collection of requirements for a paging-transmitter network has been created (for a full description please refer to [1]). They were partly expressed in form of user stories and have been divided into functional and non-functional requirements.

Main requirements are the support of the existing infrastructure (RasPager, Skyper, HAMNET) as well as a fully decentralized architecture. Everybody should be able to operate and own a transmitter or even a complete private network, having additionally the option to merge it with the
main network. Possible partitions of the network (called "split brain case") have to be recognized. The part networks have to remain their basic functionality like the transmitting of messages. After the re-establishment of the network functionality the part networks should be able to merge together and return to defined state. Furthermore, comfortable user interfaces should been available corresponding to the possible use cases.

4. Design

Various options for realizing the requirements have been considered. The first idea was a continued development of the existing Funkrufmaster. But due to the high complexity, lack of documentation, instability and the used obsolescent technologies the decision has been made to start a complete new development instead. Since probably the most practicable user interface, fulfilling the requirements, is a website, there was the approach of implementing the software as a web application using modern programming languages like "Python". Problems concerning concurrency for controlling transmitters in separated threads lead to the rejection of this option. Instead the decision has been made to use the programming language "Java", which provides the support of a lot of platforms and the availability of some needed frameworks and libraries.

4.1. Architecture

A modular software architecture has been chosen, defining three main components (see figure 3): The "transmission component" is responsible for the communication with transmitters, whereas the "cluster component" provides a reliable channel for the exchange of commands between servers. Additionally, it provides data storage and a cluster-wide synchronization. In this context a data model has been developed. Deliberately, the implementation of a (graphical) user interface was deferred. Instead, the "REST component" provides a universal interface, which offers access to all features by standardized operations, based on the concept of a "RESTful Web Service".

These three components are required to provide the main functionality of the DAPNET and together they form the "DAPNET Core" module. Additional features can be provided by other modules, which are connected to the main system over the REST interface.

4.2. Network structure

Figure 4 shows the network structure of the DAPNET. There are several instances of the DAPNET Core, which are called nodes in the following. Using the cluster component they can communicate over the HAMNET and optionally over the internet.

Transmitters are connected over a master-slave structure. Each transmitter is slave of a certain master, accordingly slave of the DAPNET Core instance, which is located closest to the transmitter. Thus malfunctions in the HAMNET will have only limited impact, which means that the DAPNET will remain at least locally its full functionality.

Since each node provides its own REST interface and all nodes are in a synchronised state it is irrelevant for the user to which node he sends his requests. This redundancy provides load balancing and a high tolerance against network errors for example a split brain case. In each network partition at least one node is reachable and therefore able to accept the user’s requests.

4.3. Cluster Component

Realizing a synchronised state between all DAPNET Core instances was the main technical problem. The first approach has been an own implementation, which has lead to a theoretical analysis of the given problem. It consists of building up a decentralized system with a peer-to-peer architecture. Equal nodes are using the shared resource of transmitters and perform as a client as well as a server. On the one hand each node can be removed at any time without limiting the functionality of the remaining ones. On the other hand at each moment new nodes can be added to the network. There are a lot of concepts and algorithms ([5], [6]) available to deal with such a network:

- Election of a leader node, which is responsible for the coordination in the decentralized system
- Algorithm for a unique leader election
- Mutual exclusion
- Synchronized clocks
- Majority decisions
- Transaction models
In order to deal efficiently with the given problem different existing solutions have been evaluated, all of them using at least some of the mentioned concepts. In the end the decision has been made to use the Java Toolkit “JGroups” [7]. It helps to build up a cluster over a network in which nodes can dynamically exit or join and reliable send (broadcast) messages. Automatically a leader is elected and crashed or not reachable nodes are recognized.

To realize a synchronized state, every request (for example from the REST component) is broadcasted to all nodes, where corresponding remote process procedure calls are executed. Each node will manipulate the local data in the same way and will perform necessary actions. If nodes are not responding they are removed from the cluster. After authentication by key and known IP address, nodes joining the cluster download the current state from another already connected node. In case of being the only node in the cluster, the data are loaded from a local JSON file. A network partition and a subsequent merge can be reliably handled by transferring the state of the largest sub-network to the smaller ones.

4.4. REST Component

The REST component offers a clearly defined interface to control all functions of the DAPNET by realizing a “RESTful Web Service” (Representational State Transfer). REST in general is an architecture for accessing resources, which follows defined paradigms (for details see [8]). Requests and responses are transferred by HTTP (for details see [9]). Data are represented with JSON.

About 20 functions are defined to access, create and modify necessary resources. One function for example allows the creation of a “call resource”, which results in transmitters sending a corresponding message. The REST component realizes a complete user and access administration as well as data validation.

The defined REST interface is the only way to control a DAPNET Core. As already mentioned, the idea is that additional modules are build on top of the interface and provide additional features, for example comfortable user interfaces (as can be seen in figure 5). Especially a web interface is demanded by the requirements, but has been excluded from this work in the first step.
4.5. Transmission Component

The transmission component implements the RasPager protocol to control the transmitters and the Skyper protocol for correct encoding of messages. To ensure that support for other pagers and transmitters can be easily added, general interfaces are used and implemented exemplary for the momentarily used hardware.

The communication and control of each transmitter is realized in a separated thread. In each prioritized queues are used. Directly before the start of an assigned time slot the fitting number of messages is forwarded from the queue to the transmitter. Time, date, rubric names and news are transmitted periodically by a scheduler.

5. Implementation

The last step of this work was the concrete implementation on the basis of the results from the design phase. In particular the non-functional requirements, like a good maintainability of the source code, a modular structure, options for monitoring the software and secure password storing were considered. Besides of JGroups the application uses a lot of other external libraries, for example “Jersey” in order to implement the REST component, “GSON” for JSON serialization, “Log4j” for logging and “Hibernate” for validation. A detailed documentation is given in [1].

First tests in an environment consisting of five DAPNET Core instances and several transmitters were successful.

6. Conclusion

The main result of this work is a fully functional software, called DAPNET Core, which is ready for an extensive test run and which fulfills all given requirements, besides of a graphical user interface. Summing up, an entire proof of concept for the given theory of realizing a paging network has been achieved. First tests of the software, based on the user stories, were already successful. Furthermore a detailed collection of requirements and informations according to the existing infrastructure has been compiled.

Assuming the support of the radio amateurs, the realization of a decentralized and modern wide area paging transmitter network could become reality.

6.1. Future Work

The next step is to launch an extensive test run in a real environment across a city-wide area. It is supposable that the software has to be optimized in respect to usage in larger networks. Lots of additional features were already requested by the radio amateurs (for example an automatic assignment of time slots to the transmitters). Assured is the development of a DAPNET Web module, which will offer the functionality of the REST interface in form of a website with a comfortable graphical user interface.

Acknowledgements

Research described in the paper was supervised by Prof. Dr.-Ing. D. Heberling, Institute of High Frequency Technology at the RWTH Aachen University. Implementation of the DAPNET Web module is done by M. Menzerath.

The authors would like to thank both for their great support and would like to announce that the complete project including the source code is expected to be released as open-source in mid 2016.

References

About Authors...

Daniel Sialkowski was born in Bochum, Germany. After earning a B.Sc. degree in Electrical Engineering, Information Technology and Computer Engineering at RWTH Aachen University in 2015, he is currently working towards a M.Sc. degree with major focus on Information and Communications Technology. His main interests are network and communication systems.

Korbinian Schraml graduated in 2013 from RWTH Aachen University as Master of Science. Since August 2013 he is a doctoral student at the Institute of High Frequency Technology. His research interest is in antennas for special applications such as radar or space-communications.

In 2010 Ralf Wilke got his diploma degree from RWTH Aachen University, Germany. Now he is working at the Institute of High Frequency Technology at RWTH Aachen University as a Ph.D. student. His main interests are satellite communication and radio networks. He is the project manager of GeReLEO-SMART, a scientific payload to be launched on the German geostationary Heinrich-Hertz satellite (H2SAT) and leader of a student group building a CubeSat communication system including space and ground segment.