PERLE: A Testbed for Pervasive Middlewares in Learning Environments

Jens Naber
University of Mannheim
jens.naber@uni-mannheim.de

Steffen Schmitz
University of Mannheim
stefschm@mail.uni-mannheim.de

Christian Becker
University of Mannheim
christian.becker@uni-mannheim.de

***Abstract*—Today's classrooms contain an increasing number of smart devices such as displays and smart lighting in the infrastructure. Additionally, many are brought by lecturers and students in the form of laptops, smartphones, or tablets. However, until now, all these devices are not fully utilized for the benefit of the lecture and a better learning experience. While the use of a pervasive middleware is the first step in connecting all the heterogeneous devices and enabling communication among them, the system also needs the right services to utilize this capability. We introduce PERLE (Pervasive Learning Environment), a testbed for pervasive middlewares in classroom scenarios. The testbed has two main goals. First, it enables the evaluation of different pervasive middlewares during lectures. Second, it shows how pervasive services enhance the learning experience of students and help lecturers in holding improved lessons. Therefore, we implemented five different services for a pervasive classroom system together with an application for lecturers and students.**

I. INTRODUCTION

Today, environments are getting increasingly smarter and people may carry multiple smart devices with them on a daily basis, e.g., smartphones, tablets, or laptops. This of course is also true for learning environments, like lecture halls or classrooms. However, while there are many devices available, they are usually not utilized to their full potential to increase the learning and teaching experience for lecturers and students. Lecturers still have to fiddle with cables and adapters to get their presentation shown on the projector and the exchange of learning material is often done via mail. Meanwhile, the content used in lectures is getting more and more digital. Multimedia material, like video or audio files, are integrated in the lesson and worksheets and student feedback are shared in real time. At the moment, this is often achieved by several independent applications and services, e.g., Apple AirPlay for video content or Whatsapp for communication and coordination, which may coexist in a classroom but do not allow for any interaction between them. Pervasive middlewares offer a solution to ease the integration of such heterogeneous devices and broad range of content without interrupting the lesson. Although the middleware handles the information exchange and device handling it needs a multitude of services to be of a profound use in a classroom environment.

PERLE (Pervasive Learning Environment) introduces a pervasive classroom system with a total of five different services, catering to support multiple tasks during a lecture. The system supports users during lectures, starting with setting up the classroom, holding the actual lesson until, finally, collecting feedback and handing out material at the end. It includes a room control service to e.g., close the shutters or dim the light. To show presentations and share multimedia content to, e.g., the classroom projector or student devices, a display service is introduced. Furthermore, the system supports the lecturer by collecting feedback in the form of surveys and further data about the students who attend the lecture, e.g., student id, major, or email. Additionally, a file transfer service is introduced, which enables the lecturer to share worksheets or further reading materials with the students or collect assignments from them. This file service is also used by other services for their file handling. Due to this extensive classroom system we are able to test pervasive middlewares in real lectures to select appropriate middlewares and fine tune their capabilities to the requirements of a teaching environment.

The remainder of the paper is structured as follows. Section II discusses related work including individual services and holistic classroom approaches. In Section III, we introduce the goals for PERLE, before discussing the architecture and design in Section IV. Afterwards, the implementation is outlined in Section V, before the evaluation in Section VI. Finally, Section VII summarizes the paper and gives a outlook.

II. RELATED WORK

There is a multitude of tools and solutions that can be used in a classroom setting to optimize learning and integrate the available devices and infrastructure into the lecture. Following, we will first take a look on single approaches related to our developed services and afterwards on complete classrooms.

Voida et al. [1] focus on different file sharing practices, like peer-to-peer sharing, e-mail, or other wiki tools. The file service presented in PERLE is close to a peer-to-peer system, but compared to existing solutions like Gnutella [2] it does not require any knowledge about the files that are offered in the network. Our file service uses an approach similar to the one in [3]. Bervens et al. divide the transfer into two parts, a metadata exchange, where the file is described, and if the user wants to receive a specific file this is handled by a direct connection between the services. Our display service builds on top of the service introduced in [4] and enables screen sharing with private and public displays. Other web-based display-services include PresiShare [5] and UBI-hotspot [6]. Those solutions require a central, public display that users connect to before they can share their content. A similar approach is used

978-1-5386-9151-9/19/$31.00 ©2019 IEEE

at the University of Mannheim with the PalMA system [7]. Manathunga et al. [8] on the other hand focus on distributing tasks and solve them collaboratively during lectures.

Writing tests and receiving feedback in classroom environments is often an asynchronous procedure. Both are often paper-based and do not allow for immediate feedback, and thus do not influence the running lecture. While there are approaches like EvaSys [9], that automate the evaluation of paper-based surveys, they are still nowhere near an instant result. Chen et al. [10] found that students prefer using a handheld device to take spontaneous tests and that it is advantageous for the lecturer to receive immediate feedback on the contents. Further, there are web-based quiz tools, like Kahoot [11]. Smart home devices like lightbulbs are increasingly common and also get integrated into conference rooms and lecture halls [12] [13]. Existing programs and utilities are tightly coupled to a given room [13] and cannot use arbitrary devices. Our room control service enables a loose coupling where all available devices are listed and exposed through a common API. The user service in our application has no direct correspondences to existing tools. Collecting and exchanging contact information usually is a manual task and either paper-based or uses shared spreadsheets like Google Docs [14].

Besides the individual services used in a classroom there are also holistic solutions that aim to provide everything lecturers and students may require in the room. Leonidis et al. [15] focus on remote, distributed classrooms and enhancing the lecturer experience by customizations of the environments. Yau et al. [16] propose solutions that are similar to PERLE. Their classroom concept uses a pervasive middleware that connects handheld devices brought by students and lecturers with the infrastructure in the room. Yau et al. focus on putting the devices at the front and center of the learning experience by communicating through them, while PERLE tries to blend into the environment without distracting the participants in the classroom. Other commonly used applications for classes include Google Classroom [17], moodle [18], or ILIAS [19]. They can be used for sharing lecture content and handing in solutions, but usually do not assist during the actual lecture.

Existing approaches try to solve specific tasks that may arise in a classroom or require a constant focus on the application at hand. Our goal is to provide a holistic solution that does not distract the lecturers and students, but instead enables a more effective and efficient lecture.

III. GOALS OF PERLE

PERLE has to serve not only as a pervasive classroom, but also as a testbed for pervasive middlewares in this particular use case. Following, we introduce a scenario for PERLE and discuss how it can be used to evaluate pervasive middlewares.

In our use case, the lecturer enters the classroom and starts the setup for the upcoming lecture. Therefore, she starts the projector, dims the lights in the room, and partly closes the shutters. Before beginning the lesson, the lecturer distributes a handout to the students, checks their attendance, and collects their e-mail addresses for further communication after the class. Afterwards, she starts her presentation on the main projector. If the students are interested in the slides they have to wait until they receive them via e-mail. During the lesson, the lecturer may change from a one-way presentation to a group project. After the group phase, the students come to the front and present their results. Therefore, they have to bring their laptops and connect them to the projector. After the lesson, the lecturer wants to collect the feedback of the students via a survey. The survey results have to be digitalized for further processing. At the end, everything needs to be shut down and back in the office the lecturer has to distribute the lecture slides to the students.

The goals of PERLE include a *comprehensive support* of the given scenario. The lecturer and students should be supported by the PERLE throughout the complete lesson. This includes controlling the devices in the room, presenting during the lecture, the exchange of data and personal information, and giving feedback to the lecturer. Further, the services should be *encapsulated*, to allow for multiple possible combinations of services and different pervasive middlewares. For instance, if a survey is not required in the given use case it could be left out, or the testbed can be extended with more services, if further functionalities are needed. Additionally, the pervasive middleware should be easily exchangeable to allow to test a wide range of middlewares.

On top of the requirements aimed directly at PERLE, the testbed should also be able to evaluate relevant features of the underlying pervasive middleware in future field tests. Lectures may be in changing rooms and different combinations of students and lecturers may be present. Due to this, the middleware should be able to form a *distributed and ad-hoc* system. This eliminates the need for a central instance and it does not rely on the laptop of the lecturer, or if a stationary computer is available. To evaluate this, all services in PERLE should be distributed and may not rely on central servers. Further, the middleware should be *scalable*, as the lecture audience may vary between just a few to several hundred students. The testbed should be easily deployable on as many devices as necessary and include performance measurements. Lastly, the pervasive middleware should guarantee a high *interoperability*. Therefore, it has to support a large heterogeneity on a device level. This includes different device types, like stationary PCs or tablets, but also different operating systems.

IV. PERLE TESTBED

In the following section, we will first discuss the overall architecture of PERLE. Afterwards, each of the developed services for our testbed will be explained in more detail. This also includes the interaction between services and the applications for the lecturer and students.

A. General Architecture

PERLE utilizes a pervasive middleware to connect the infrastructure devices in a classroom and the devices brought by end-users. We differentiate between devices used by the lecturer, e.g. a stationary PC or her tablet, and student devices,

like laptops or smartphones. PERLE aims at supporting the overall lecture and enhancing the experience of the students, as well as the lecturer. Therefore, we developed a total of five different pervasive services:

- File Transfer Service
- Display Service
- Survey Service
- Room Control Service
- User Service

Additionally, we introduce an application for lecturer and students, which aggregates the functionalities of the services and allows the users to interact with them in an easy way.

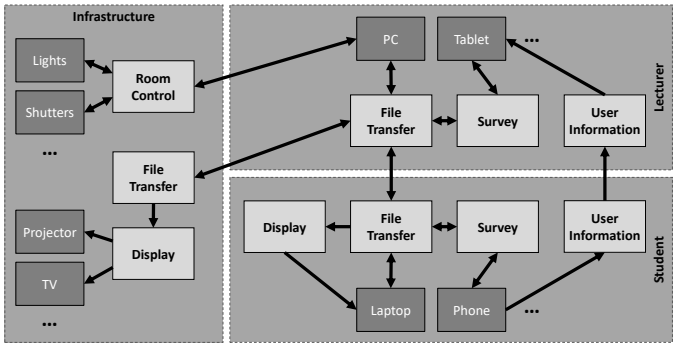

Fig. 1. Overall architecture demonstrating the information flow between different device types (dark gray) and the services (light gray).

Figure 1 shows the general architecture of PERLE and the communication between the different services and devices in the environment. Each of the services can run in several parts of the environment, e.g., the display service can be executed on the projector in the classroom infrastructure and on the laptops of students to show the presentation on all devices. The file transfer service notably is running on all devices and not only offers its functionality to exchange files to the end-user, but also allows other services to distribute their data. This, for instance, enables the display service to retrieve presentations from the file transfer service.

B. Services

In the following, we will discuss each of the services in more detail. Therefore, we will first take a closer look on its goals and how it fits into the scenario described in Section III. Afterwards, we will take a closer look on the design specifics.

File Service: The file transfer service has two main functions. First, it allows students and lecturers to exchange files between each other. Second, it can be used by other services for their internal data handling. This allows, for instance, the lecturer to distribute the presentation slides to the students, and also the display service to listen for published presentations and show them on the projector. The file transfer service is therefore an essential link in the classroom and the only service to be used by other services.

The file transfer service is designed as a completely distributed peer-to-peer system. Each device must maintain a local index of shared files. Applications on this device can add and remove files to this index. For each file the file type, name, and path is stored together with an unique ID. The index can be requested from other devices in the pervasive system. The service returns a list with the metadata of all files together with its IP and the used port. Applications are able to receive a complete view of all available files in the pervasive system by collecting all local indexes. The file transfer service enables users to share files, but is also used internally by other services. For device and service discovery the underlying middleware is used. The actual data transfer is not handled directly via the pervasive middleware. If the client is interested in one or more of the files provided via the service a TCP socket is opened between them. The client then requests a specific file via its ID and the actual data transfer is then handled via the direct connection. This reduces the load on the middleware and the file transfer service is not dependent on support of the middleware for every possible file type.

Display Service: The display service [4] previously developed in our group was further enhanced to be used in pervasive classroom scenarios. It is not only included in the applications for lecturers and students, but is also able to run directly in the classroom infrastructure and enables the utilization of projectors or public displays through the pervasive middleware. The service enables the users to share multimedia content on screens in the classroom, but also directly with the personal devices of other users. This allows, for instance, the lecturer to show her presentation on the projector and simultaneously on the devices of students. Further, students would be able to mirror their laptop screen to the lecturer or projector, if they have questions on their local content.

As discussed in [4], the display service allows for sending single images and text directly via the underlying pervasive middleware. Additionally, it is possible to send a continuous stream of images. Therefore, only the IP address of the display service is exchanged via the middleware with the client. The actual stream is then realized directly between the two peers as a TCP based stream. This reduces the overhead of the communication through the middleware. Additionally, the display service is able to show presentations in the PDF file format. Therefore, the PDF file is transmitted to the client and split up into single slides, which are saved as images. This eliminates the need to distribute each slide to all devices on every slide change. After the initial transmission of the file, the service is only notified if it has to switch slides, reducing the required bandwidth and latency. The display service can be executed standalone, for instance, directly on a public display.

Survey Service: During or after a lesson the lecturer may want to collect feedback or conduct short questionnaires to test the student's understanding. Today, these surveys are often paper-based. With the introduction of our service the lecturer is able to share her questions directly with the students via the pervasive middleware and collect the responses. This decreases the complexity of handing out the survey and the need to digitalize and evaluate the handwritten responses.

The surveys are predefined by the lecturer and saved as JSON files. This enables lecturers to use surveys designed

by others or to introduce a database of surveys in schools or universities. While it is possible for the lecturer to write the survey in the JSON format, an external graphical tool with JSON export may be used. The actual tool for creating these surveys is not part of this paper. Lecturers are able to select a questionnaire in the application, which is distributed to the students. If a file transfer service is present in the pervasive system it is used for the distribution process. If none is present the questionnaire is send to the survey service on each student device individually. Afterwards, the service on the lecturer device waits for incoming responses from the students. When the lecturer decides she has enough responses from the students she may stop the survey. Afterwards, the results are saved to evaluate them later.

Room Control Service: There are many devices in a classroom which may not directly be used for the teaching process. Nonetheless, they need attention by the user. This includes controlling the shutters or lights in the room, or turning on all displays and projectors before the start of the lesson. The goal of the proposed room control service is to make these devices accessible through a pervasive middleware. This would not only enable the control of these devices through one unified application, but also to make use of sensors in the pervasive system for proactive adaptation.

The room control service acts as an intermediate between the pervasive middleware and typical smart devices from the home automation market, like smart light bulbs or automatic window shutters. It, therefore, makes use of the REST API provided by many of these smart devices, e.g., as by Nest[1]. If additional devices without REST API should be supported it would be possible to extend the service with further APIs. The room control service offers a list of all available devices together with their actual status via the pervasive middleware. Applications in the system are then able to toggle the status of the devices or set specific parameters. The service can be run once in the system's infrastructure and connect to all available and REST enabled devices, or it can be run several times e.g., for each device which should be controlled. Therefore, there is no need for a central instance and this service can also be deployed in a distributed fashion. The application afterwards will be able to contact all instances of the service in the system and collect the complete list of smart devices.

User Service: To allow students and lecturers to exchange personal information in the classroom we developed a user service. Today, lecturers often depend on paper lists to collect information about the audience. This could be simply a signature list to check if every registered student is present, or the email contacts to send further reading material after the lecture. To remove this tedious process from the lesson we introduce the user service, which handles the information exchange between students and lecturers.

The user service combines the advantages of a persistent database with the use case of a distributed and ad-hoc pervasive network. At the first start of the service on a device the users have to register with their personal information. In the current implementation, this includes the full name, e-mail address, and a password. This could be easily extended, e.g., with current major, completed courses, or matriculation number. Additionally, the service stores the role of the user, e.g., student or lecturer. With the role it is possible to adapt the offered functionalities in applications to the needs of the user. After the registration, the local instance of the user service saves these information into a persistent SQLite database. Due to this, the students only have to enter their information once. After establishing the ad-hoc pervasive system in the classroom the user service searches for an instance of itself running on a lecturers device. It then sends the information stored in its local database to this service. On the lecturer device all incoming information are also stored in the local SQLite database. This allows the lecturer to look up who was attending the lesson back in the office after it ended. Additionally, the contact information of the students are automatically exchanged, which allows for an easy distribution of further material or information after the lecture.

C. Student and Teacher Application

To aggregate the services and make them usable for lecturers and students we developed a shared application. While the main structure is identical for both groups, the offered functionalities may differ for each role. To ease the development, ensure a similar experience for all users, and minimize training effort for users, we developed a single application, which is automatically adjusted based on the users role. Students are able to use the application to follow the lecture slides, share their screen with the lecturer, upload and download files, and fill out surveys. Lecturers are additionally able to view contact information, control smart devices in the room, distribute a survey, and start, stop, and switch the slides of a presentation.

The application is split into three major user interface parts. The middle part of the window represents the main application and menu, the outer two parts are only shown if necessary and display content or service functionalities. This helps to keep the application clean and only provide information and functionalities applicable to the current situation. The middle main window of the application contains an area for quick actions, which aims to bundle functionalities of the pervasive classroom and gives an easy and fast access to often needed services. These could be to just download a newly added file from the file service, or more complex tasks involving several services. This could be, for instance, used to set up the room for a lecture with one click (start the projector, dim the lights, and share a presentation). Lecturers are able to create these quick actions beforehand and save or share them to be used during the lesson. Additionally, the main window offers a menu to access the individual services and some often used functionalities of the room control service.

The left part of the application expands if a specific service is selected through the main menu. It then shows all functionalities of this service available to the user, for instance, all devices in the room and their functionalities. The

[1]https://developers.nest.com/guides/api/rest-guide

right part of the application is needed to show content to the user. This is mainly used by the display service, which allows the lecturer to show the presentation directly on the student devices. Additionally, students are also able to use the display service to share their screen, if they have problems or questions. This shared screen would then appear on the right side of the lecturers application.

V. Implementation

The PERLE testbed is implemented in Java and the prototype utilizes the BASE middleware [20] for device handling, service discovery, and communication. All services included in PERLE are encapsulated and aggregate all calls to the middleware API in a single exchangeable class. To migrate between different middlewares only these classes need to be altered. Additionally, it is possible to provide classes for multiple middlewares and change the underlying middleware quickly during an evaluation phase. For future evaluations we included the possibility to conduct several performance measures. So, PERLE, for instance, protocols the execution times of different middleware calls and the data received by the middleware like available devices or services.

Further, we utilize the iCasa simulator for pervasive smart environments [21]. Due to this, we eliminate the need to set up different smart enabled devices for the testbed. Within the simulator, it is possible to create the layout of the pervasive classroom and add smart devices, like light bulbs, shutters, or projectors. The iCasa simulator offers a REST API to receive the available devices and communicate with them. Our room control service accesses the simulator via this API and offers the functionalities of the devices to the user. Additionally, the REST API allows us to switch easily between the simulator and REST enabled devices in a real classroom.

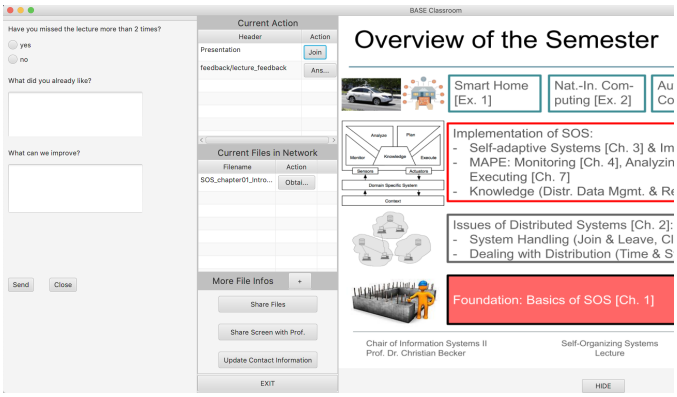

Fig. 2. Screen shot of the application with the three main user interface parts.

Figure 2 shows the user interface of the application for lecturers and students. The screen shot illustrates the view of the student. For the implementation JavaFX was used, which allows for a very dynamic and extensible layout of the interface. The aforementioned division in three major parts can also be seen in the image. With this modular layout it is easily possible to extend the application for further services. It only needs to be included in the main menu in the middle of the application and it has to provide a JavaFX pane to offer its functionalities. This pane is then dynamically shown as needed in the left or right part of the application.

VI. Evaluation

In the following section we discuss the goals set in Section III. Further, we conducted expert interviews with lecturers to evaluate if it is possible to augment real lessons and improve the learning process with PERLE.

We introduced five different goals for the PERLE testbed. To *support the scenario* introduced in Section III we developed five different pervasive services together with an application for end users, which aggregates all functionalities. During the development process we made sure, that all calls to the pervasive middleware are *encapsulated* for each service. This enables us to easily switch out the underlying pervasive middleware during the evaluation with the testbed. Additionally, all services are implemented without the need for a central server or storage. While the file transfer service can be used by the other services to simplify their data handling, it is not required. This allows for different configurations of the testbed and fulfills the goal for a *distributed* system. Further, the testbed is able to evaluate different characteristics of pervasive middlewares. This includes the possibilities to create *ad-hoc* pervasive classrooms without a centralized instance. Additionally, the testbed is able to protocol the delays introduced by the pervasive middleware and thus enables to compare the *scalability*. Further, due to the implementation in Java it is possible to migrate the testbed for multiple platforms, like Windows, MacOS, or Android. This supports the goal of *interoperability*. Figure 3 shows an example setup of the PERLE testbed in a classroom scenario.

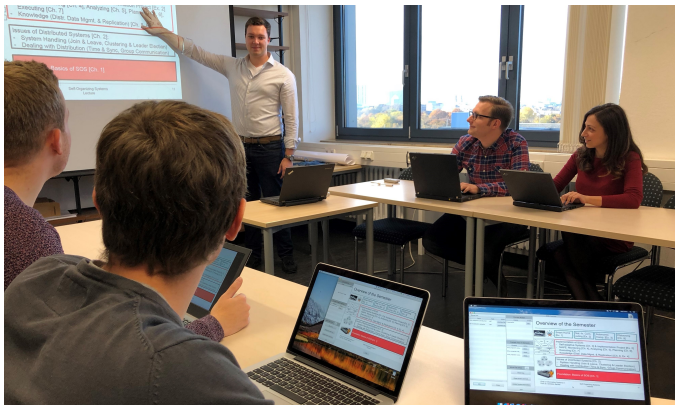

Fig. 3. An example setup of PERLE in a small lecture room.

To evaluate the usability of our prototype we conducted expert interviews. We gave the PERLE application to a total of ten participants, who tested it either as a lecturer or student. Afterwards, they answered a questionnaire. The questions mainly concerned the ease of use and usefulness of the applications and services overall and for specific features rated

on a scale from 1 (strongly disagree) to 5 (strongly agree). Additionally, we gained valuable insights through open questions regarding the expected functionalities and occurred problems while using PERLE. The participants evaluated the overall usefulness of the PERLE application with an average score of 4.26 and thought with a value of 4.7 that the testbed includes all functionalities necessary for the use in a classroom. Further, the users said with an average score of 4.3 that they enjoyed using the application. When looking at specific features the most positive feedback was for the quick actions. The users rated the helpfulness of them with an average score of 4.5. Participants using the application as a lecturer rated it with a straight 5 compared to 4.16 by users participating as students. Additionally, they were asked if these quick actions led to a faster work flow, which was rated with 4.4 (lecturers 4.75 and students 4.0). This result is a hint that the quick actions are at the moment more tailored to the need of the lecturer. The lowest scores of the interviews were for the user interface, where the overall user friendliness was rated with an average score of 3.89 and the ease of navigation the application with 3.99. While the score is still quite high, it corresponded to the insight gained by the open questions. Here, the most suggestions by the participants were related to the terminology used to describe the functionalities and the naming of devices in the room control service. Overall, the user interface should be less technical to support lecturers and students in non-IT related fields. Additionally, participants asked for more predefined quick actions, with regards to starting and ending a lecture or controlling several devices in the room at once.

As a result of the expert interviews we are able to refine the application further and increase the usability for all users. Overall, the evaluation showed that we are on the right path and that PERLE can be of great use in classrooms. Therefore, we plan large scale user studies and evaluations during real lectures in the near future.

VII. CONCLUSION

Over the course of this paper, we introduced PERLE, our testbed for pervasive middlewares in classrooms. It allows us to evaluate different middlewares on their performance and usefulness in pervasive classrooms. In total, we developed five different services, which are able to interconnect the present devices and enhance the lessons. We are able to support the lecturers and students in different phases of the class, by offering services to share personal information and files, show content on projectors and displays, control smart devices, and conduct surveys. Together with the introduced application for lecturers and students we aim to simplify repetitive tasks, which distract from the actual goal of an engaging and educational teaching experience. In expert interviews, we could see that lecturers are fond in using such a pervasive classroom. Further, the testbed is able to protocol the performance of the pervasive middleware and therefore compare different middlewares regarding their usability in classrooms.

In the near future, we plan to extend the PERLE testbed further. We plan a service for direct feedback to the lecturer during the lesson. It should enable students to report if the lecturer is, for instance, too quiet or too fast. The lecturer receives feedback immediately as a notification and can react instantly. Further, the next step will be to roll out the testbed in real pervasive classrooms and start with large field tests. The two main goals of these studies will be to first evaluate the performance of different middlewares in a classroom use case and second gain insights in the usability of PERLE during lectures. We are planning for several field tests with a scale of 20 to 100 students.

ACKNOWLEDGMENT

We would like to thank our students Dennis Knödler, Roman Scholz, Nadja Seemann, and Alexander Tsarov for their contributions.

REFERENCES

[1] S. Voida, W. K. Edwards, M. W. Newman, R. E. Grinter, and N. Ducheneaut, "Share and share alike: exploring the user interface affordances of file sharing," in *Proc.CHI*, 2006.
[2] M. Ripeanu, "Peer-to-peer architecture case study: Gnutella network," in *Proc. P2P*, 2001.
[3] W. A. Berkvens, A. Claassen, J. P. Van Gassel, and A. Sinitsyn, "Media distribution in a pervasive computing environment," in *Proc. PERWARE*, 2005.
[4] J. Naber, D. Schäfer, S. VanSyckel, and C. Becker, "Interactive display services for smart environments," in *Proc. PICOM*, 2015.
[5] M. Geel, D. Huguenin, and M. C. Norrie, "Presishare: opportunistic sharing and presentation of content using public displays and qr codes," in *Proc. PerDis*, 2013.
[6] T. Ojala, H. Kukka, T. Lindén, T. Heikkinen, M. Jurmu, S. Hosio, and F. Kruger, "Ubi-hotspot 1.0: Large-scale long-term deployment of interactive public displays in a city center," in *Proc. ICIW*, 2010.
[7] "Palma university of mannheim, URL:https://github.com/ub-mannheim/palma. access: 29.10.2018."
[8] K. Manathunga, D. Hernández-Leo, J. Caicedo, J. J. Ibarra, F. Martinez-Pabon, and G. Ramirez-Gonzalez, "Collaborative learning orchestration using smart displays and personal devices," in *Design for Teaching and Learning in a Networked World*, 2015.
[9] "Evasys, URL:http://en.evasys.de/main/products/survey-and-evaluation-software/evasys-suite/evasys-education.html. access: 29.10.2018."
[10] F. Chen, B. Myers, and D. Yaron, "Using handheld devices for tests in classes," Carnegie Mellon University School of Computer Science, Tech. Rep., 2000.
[11] "Kahoot, URL:https://kahoot.com. access: 29.10.2018."
[12] A. Waibel, T. Schultz, M. Bett, M. Denecke, R. Malkin, I. Rogina, R. Stiefelhagen, and J. Yang, "Smart: The smart meeting room task at isl," in *Proc. ICASSP*, 2003.
[13] L. Harper, A. Gertner, P. Herceg, T. Hines, E. Kemon, D. Mireles, M. Shadid, and J. Van Guilder, "Pervasive and ubiquitous computing in a corporate conference room," 2004.
[14] C. Kongchan, "How edmodo and google docs can change traditional classrooms," in *Proc. ECCL*, 2013.
[15] A. Leonidis, M. Antona, and C. Stephanidis, "Enabling programmability of smart learning environments by teachers," in *Proc. DAPI*, 2015.
[16] S. S. Yau, S. K. Gupta, F. Karim, S. I. Ahamed, Y. Wang, and B. Wang, "Smart classroom: Enhancing collaborative learning using pervasive computing technology," in *Proc. ASEE*, 2003.
[17] "Google classroom, URL:https://classroom.google.com. access: 29.10.2018."
[18] "Moodle, URL:https://moodle.org. access: 29.10.2018."
[19] "Ilias, URL:https://www.ilias.de. access: 29.10.2018."
[20] C. Becker, G. Schiele, H. Gubbels, and K. Rothermel, "BASE - a micro-broker-based middleware for pervasive computing," in *Proc. PerCom*, 2003.
[21] P. Lalanda, C. Hamon, C. Escoffier, and T. Leveque, "icasa, a development and simulation environment for pervasive home applications," in *Proc. CCNC*, 2014.