TIROL: The Extensible Interconnectivity Layer for mHealth Applications

Christoph Stach, Frank Steimle, and Ana Cristina Franco da Silva

University of Stuttgart, Institute for Parallel and Distributed Systems,
Universitätstraße 38, 70569 Stuttgart, Germany
stachch@ipvs.uni-stuttgart.de
steimlfk@ipvs.uni-stuttgart.de
francoaa@ipvs.uni-stuttgart.de

Abstract The prevalence of various chronic conditions is on the rise. Periodic screenings and a persistent therapy are necessary in order to aid the patients. Increasing medical costs and overburdened physicians are the consequences. A telemedical self-management of the illness is considered as the answer to this problem. For this purpose mHealth applications, i. e., the synergy of common smartphones and medical metering devices, are vitally needed. However, poor device interoperability due to heterogeneous connectivity methods hamper the usage of such applications. For this very reason, we introduce the concept for an ex**T**ensible **I**nte**R**c**O**nnectivity **L**ayer (*TIROL*) to deal with the interconnectivity issues of mHealth applications. Furthermore, we present a prototypical implementation for TIROL to demonstrate the benefits of our approach.

Keywords: mHealth, medical devices, harmonization, interconnectivity layer.

1 Introduction

Today, smartphones became constant companions for almost everybody. Due to their consistent connection to the Internet and their ever increasing battery capacity, they serve as a permanent information source as well as communication tool. However, the most outstanding feature of these devices is that any developer can provide new applications for them. Since the number of built-in sensors raises with each smartphone generation, the application possibilities for smartphone are virtually unlimited. Even ordinary smartphones contain hardware required for e. g., basic health and wellness tracking. Moreover, additional sensors tailored to special use-cases can be connect to smartphones, e. g., via Bluetooth.

As a consequence, it is hardly surprising that, especially in the health sector, the use of smartphones can be highly beneficial in terms of saving treatment costs and helping patients who cannot visit their physicians regularly [17]. Patients at any age benefit from health applications for their smartphone—the so-called *mHealth applications* [16]. Particularly advantageous is the telemedical treatment of chronic disease such as diabetes mellitus or the chronic obstructive pulmonary

In: Damaševičius R. and Mikašytė V. (Eds.) Information and Software Technologies. ICIST 2017. Communications in Computer and Information Science, vol 756. Springer, Cham, pp. 190–202, 2017.

disease (short *COPD*). While hitherto an episodic care in a clinic is required for chronic diseases, mHealth is an enabler for home healthcare. With the help of a smartphone, medical metering devices, and proper mHealth applications, patients are able to diagnose, screen, and therapy their disease by themselves and have to visit their physicians only in case of an emergency, since the required measurements can be carried out very easily with affordable medical measuring devices [24]. That way, not only the physicians' workload gets reduced both also insurance companies safe a lot of money.

However, in order to operate hitch-free, mHealth applications need to be compatible to any available medical metering device, i. e., the application running on a smartphone has to be able to interchange data with the (mostly) external medical hardware. The data interchange constitutes a crucial weak spot for mHealth due to heterogeneous connection types and non-uniform communication protocols [1]. Users are repelled by device incompatibilities, that is, every application supports certain devices only and in some circumstances users have to own several similar medical devices to be able to use all of their mHealth applications. As a consequence, mHealth does not get unreserved approval by the patients despite all of its unquestionable benefits [13].

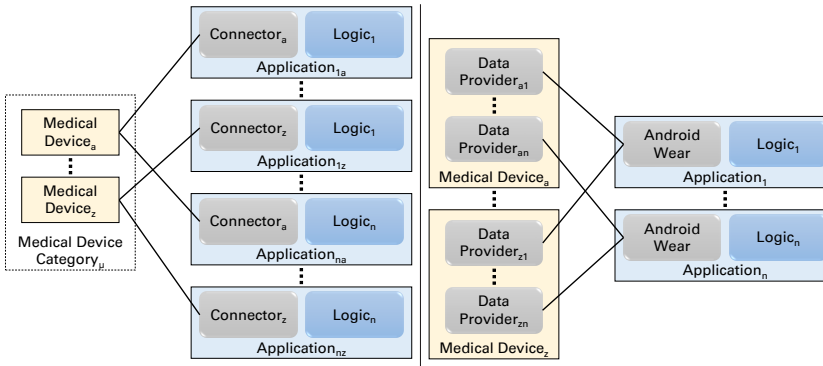
On that account, this paper provides the following key contributions:

(1) We postulate a requirements specification for an interconnectivity layer which harmonizes the connection techniques.
(2) We introduce the concept for an ex**T**ensible **I**nte**R**c**O**nnectivity **L**ayer (*TIROL*) to deal with the interconnectivity issues of mHealth applications.
(3) We implement TIROL as an extension for the **P**rivacy **M**anagement **P**latform (*PMP*) [20, 21].
(4) We assess the utility of TIROL.

The remainder of this paper is structured as follows: Sect. 2 details on the state of the art concerning connection techniques used by medical metering devices and Sect. 3 looks at related work dealing with the interconnectivity issues of mHealth applications. Based on the shortcomings of these approaches, Sect. 4 postulates a requirements specification for an interconnectivity layer which harmonizes the connection techniques. Following this, the conceptual specification for TIROL is introduced in Sect. 5 and Sect. 6 gives some insights on the implementation of TIROL. Sect. 7 assesses whether TIROL fulfills the requirements specification. Finally, Sect. 8 gives a conclusion and a brief outlook on future work related to mHealth applications.

2 State of the Art

Basically almost every modern medical metering devices in home healthcare is a standalone system. Yet, these devices are designed mainly focusing on their primary task—the metering of health data—and the visual processing of the data is often missed out. Here smartphones come into play. They can be used for both, as a guidance of how to perform a metering as well as a comprehensible



(a) Standard Bluetooth Connection (b) Wear-Based Connection

Figure 1: Comparison of Standard Bluetooth and Wear-Based Connections

presentation of the results [16]. Nowadays the connection is usually realized via Bluetooth in order to ensure a user-friendly operation. However, concerning the communication protocols such a de facto standard is not in sight. There are several proposals for a uniform connection type, but none of them prevailed yet.

The Bluetooth SIG came up with the *Bluetooth Health Device Profile* (*HDP*) [6] in order to supersede the outdated *Bluetooth Serial Port Profile* (*SPP*) as the SPP causes a poor level of interoperability with different medical devices. The HDP was adopted in 2008 as part of the ISO/IEEE 11073 family of standards. Its main focus is to support a variety of applications for home healthcare. For that purpose, a new communication protocol is introduced, the *Multi-Channel Adaptation Protocol* (*MCAP*). The MCAP is responsible for a reliable connection between a data source (i. e., a medical device) and a data sink (i. e., a smartphone or a PC). Initially, it establishes a control channel between source and sink, e. g., to synchronize the two devices. Then, the devices are able to open multiple channels for payload data. This mode of data transmission is optimized for devices with low resources, as medical devices often have limited computational power. The Bluetooth SIG provides furthermore an optimized exchange protocol to enable interoperability between sensors and data management devices [7]. This includes device data specializations which define how certain families of medical devices (e. g., glucose meters or peak flow monitors) provide their data. I. e., the Bluetooth SIG specifies service IDs and data models for these families. Whereas it is highly recommended that vendors of medical devices should comply with these specifications, it is no necessity for the HDP. As a result, developers of mHealth applications cannot rely on these specializations and they still have to create multiple versions of their applications in order to support medical devices from different vendors (see Figure 1a). Although each version differentiates only in the connector component, the HDP is still no enabler for mHealth interoperability.

Moreover, HDP belongs to the Classic Bluetooth profiles, i. e., it does not support *Bluetooth Low Energy* (*BLE*). BLE is designed primary for smart devices

where low power consumption is crucial. Due to the new introduced Low Energy protocol stack, BLE is able to reduce power costs considerably. Contrary to Classic Bluetooth, in BLE the devices remain in sleep mode most of the time and initiate a connection only as long as data needs to be transferred. Especially in an mHealth setting where only small amounts of data have to be exchanged and the interval between two measurements (and thus data transfers) is relatively long, BLE is particularly favorable [4]. Thus, a lot of novel medical devices prefer BLE over Classic Bluetooth in order to prolong its battery life. As BLE specifies the communication protocol only but neither standardized device IDs nor message formats for the data exchange, similar medical devices from different vendors establish heterogeneous proprietary communication norms. The Personal Connected Health Alliance consortium promotes various health care profiles in order to counteract this fragmentation [2]. However, the number of proprietary communication protocols such as *Terminal I/O* [23] does not decrease sustainably.

Android Wear is an Android version designed for accessing sensors in wearables. Android Wear facilitates the pairing of wearables with smartphones. For this purpose, it provides a generic API to the sensors of the wearable. This API abstracts from specific communication techniques and protocols. In order to be able to use this API, an mHealth developer has to implement a data provisioning component for each and every of his or her applications. The provisioning component is installed on the wearable. It is needed to establish the connection to the smartphone and it prepares the sensor data for the transmission. This implies that each mHealth application on the smartphone needs a counterpart running on the wearable (see Figure 1b). Thereby not only the storage consumption but also the power consumption of the wearable increases. Moreover, Android Wear is only compatible to a few certified wearables [12]. Thus, this is also not a comprehensive solution for the interconnectivity issues of mHealth applications.

While most modern medical metering devices in home healthcare have a Bluetooth interface, some vendors still rely on tethered technologies. E. g., the *PO 80* by Beurer Medical[1] has an USB port, only. Other devices such as the *iHealth Vista* by iHealth Labs[2] sends any captured data to an online health service provider. This provider preprocesses the data and makes it available via a web-based interface and / or via APIs for third-party RESTful applications. As a consequence, the already heterogeneous connectivity landscape for medical metering devices gets even more fragmented.

Since it is not conceivable that any of the available connection methods or communication protocols prevail anytime soon, both, developers and users are in great want of a reasonable solution for this fragmentation issue. Currently developers are forced to implement their applications several times in order to support various devices. This leads to long waiting times for patches, since each update has to be applied to every version of the application. Needless to say, that no application is compatible with every device and therefore, users gets furthermore frustrated. That is why also research projects deal with this issue.

[1] see www.beurer.com/web/us/products/pulseoximeter/pulse_oximeter/PO-80
[2] see www.ihealthlabs.com/wireless-scales/wireless-body-analysis-scale

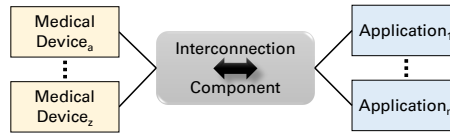


Figure 2: Hub and Spoke Architecture Realized by a Connection Interlayer

3 Related Work

The state of the art approaches show, that the solution for the interconnectivity issues of mHealth applications cannot lie in the definition of another standard. Rather promising is the introduction of a unifying interlayer realizing the interconnection between medical devices and mHealth applications [18]. Instead of the huge number of connectors or data providers required per mHealth application (see Figure 1) a connection interlayer establishes a hub and spoke architecture (see Figure 2). This reduces the number of required connections sustainably.

The *Data Management System* [14] introduces a layer architecture gathering data from input streams (e. g., medical devices) and sending this data to output streams (e. g., smartphones). The focus of this work is on input stream validation and data consistency. For establishing the data streams—and therefore the connection of medical devices—an agent middleware is required which is not included in the approach. Masaud-Wahaishi and Ghenniwa [11] introduce a brokering layer connecting service providers (i. e., data sources) with service requesters (i. e., applications). The brokering layer however does not focus on the connection itself, but on the privacy-aware processing and provisioning of data.

The *Mercury* system [9] deals with this problem by setting up a wireless sensor network for all health sensors. This network provides a single and unified access point for applications. However, specialized microcontrollers are required in the sensors to join the network. Likewise Otto et al. [15] suggest to use wireless body area networks to connect medical devices with smartphones. The smartphone can be used as both, a runtime system for applications as well as a connection to the Internet. Yet, this approach also requires particular hard- and software.

Kouris and Koutsouris [8] recommend to use IoT techniques for the data transfer. Therefore, all sensors are connected to the Internet and send their data to a cloud-based database. mHealth applications can access the sensor data via this database. Even though there are secure transfer protocols for health data such as *Hide-n-Sense* [10], a permanent and unrestricted transmission of sensitive health data to an unknown server is ineligible for most users.

Sorber et al. introduce a vision for a central component called *Amulet* harmonizing the communication with any kind of medical device [19]. Moreover, the Amulet is able to connect to a smartphone, either for providing application with health data or for sending the data to online health services. Amulet should have a small form-factor so that patients are able to carry it always along. This vision is realized as part of the Amulet platform [5]. However, the platform requires a proprietary firmware for the sensors alongside with a distinct runtime-system

for the applications. Stock devices are not supported. *BodyScan* [3] has a similar strategy. Their so-called *Wearable Platform*, i. e., the central component of BodyScan, is additionally able to analyze changes on the radio signals of the medical devices. Via these signal changes BodyScan is able to recognize activity pattern in order to enrich the gathered data. Nonetheless, BodyScan has the same disadvantages as Amulet, namely the need for a particular hardware.

As none of these approaches is outright promising, we postulate a requirements specification for an mHealth interlayer in the next section.

4 Requirements Specification

Based on the features as well as shortcomings of the state of the art approaches and related work towards a solution for the interconnectivity issues of mHealth applications, we devise the following requirements which are vital for a unifying interlayer that deals with these problems:

(*A*) **Heterogeneous Connectivity.** As shown above, mHealth applications are executed in a heterogeneous environment. In order to homogenize the environment, the interlayer has to able to support various connection types (such as Classic Bluetooth, BLE, or tethered connections) as well as communication protocols (such as HDP or Terminal I/O).

(*B*) **Flexible Connectivity.** Similar to Android Wear, the interlayer has to abstract from a certain hardware. From an mHealth application's point of view, it is irrelevant which medical device provides a certain kind of health data. Therefore, the devices can be bundled into categories characterized by their features (e. g., respiratory devices or cardiac devices). In this way, an application has to address its data request to a certain category only instead of a certain device and the interlayer forwards it to the right device.

(*C*) **Extensible Interface.** The mHealth landscape is constantly evolving, i. e., new connection types, communication protocols, health devices, and even device categories emerge consistently. Because of that, the interlayer has to be extensible. Otherwise it would be deprecated and thus useless in no time.

(*D*) **Abstract Interface.** The interlayer should provide an abstracted interface to the medical devices. I. e., an application developer has just to define which data s/he wants to access and the interlayer arranges the access.

(*E*) **Resource-Efficient Data Access.** In an mHealth environment, a long battery life is a key issue. Therefore, the data access by the interlayer has to be very resource-efficient. First and foremost, it has to be prevented that every application has to install its own connector on every medical device, as it is needed for Android Wear.

(*F*) **Reliable Data Access.** The interlayer should be able to validate the transmitted values and counteract transmission failures.

(*G*) **No Dependencies.** The interlayer must not require any special hard- or software. In particular, a manipulation of the health device (e. g., by installing an additional transmission module) has to be preempted by any means, due to safety issues. Such an interference could result in corrupted data.

5 Conceptual Specification

On a conceptual level, our proposed ex**T**ensible **I**nte**R**c**O**nnectivity **L**ayer (*TIROL*) is located in between the application layer which executes the applications and the hardware abstraction layer which manages the devices (see Figure 3). So, mHealth applications are completely independent from the devices. TIROL offers them data domain specific interfaces (e. g., respiratory data or cardiac data) to the devices. The key characteristics of TIROL are detailed in the following:

TIROL is extensible due to a dynamic plugin system towards the hardware abstraction layer. Each plugin represents a certain data domain. It specifies which data is available in the respective domain. This specification is completely device independent—i. e., it uses a uniform data format defined by TIROL—and it abstracts from a specific access procedure of the devices. Each plugin offers only data from its domain. If a health device offers data from multiple domains (e. g., the peak expiratory flow *[respiratory]* heart rate *[cardiac]*), then two separated plugins are responsible for that device.

The plugins can be added or removed at any time. This is necessary since some data is only temporarily available. E. g., some medical devices are turned on only to perform a metering, to broadcast the results, and are shut down subsequently. A patient is also able to add new device types to TIROL, e. g., when his or her physician wants to monitor additional health values. Any application has then automatically access to the newly added data.

Additionally, each plugin is also extensible due to a flexible connection interface. While the plugin itself represents the interface towards the applications, this connector is the interface towards the devices. The connector manages the connections to the devices including the individual connection types and communication protocols. Since a patient might have several devices of a certain device type (e. g., a smart watch and a heart monitor that provide heart rate data), the connection interface has to be able to exchange the connected device at runtime. Thereby it is always possible to select the device with the best accuracy which is currently available. Whenever a new connection type or protocol evolves, it is only necessary to adapt the connector and afterwards devices uses this new connection technique are instantly available to any application.

TIROL also reduces the number of required components for the connection of health devices and applications. While the state of the art approaches require $a \times d$ connecting components[3] either on the application side (see Figure 1a) or on the device side (see Figure 1b), our approach gets by with h plugins and m extensions for the connection interface per plugin—i. e., $\sum_h m \in h \approx d$[4]. In other words TIROL requires approximately one connecting component per device, no matter how many applications are used. Moreover, since the interlayer harmonizes the heterogeneous connectivity landscape via these connecting components (plugins as well as connection interfaces), no special hardware or software is required. I. e., any standard device and application can be connected to TIROL.

[3] Let a be the number of application and d the number of devices.
[4] Let h be the number of health domains and m the number of device models.

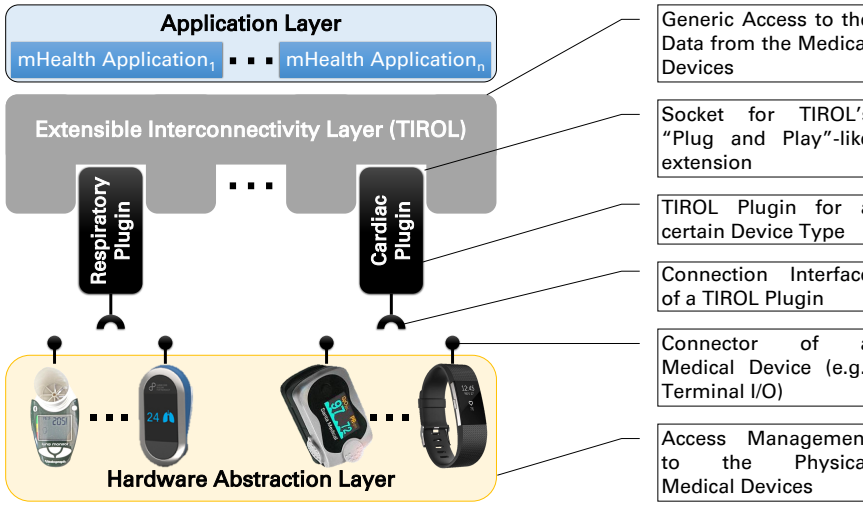


Figure 3: Conceptual Design of the Extensible Interconnectivity Layer (TIROL)

That way, TIROL is very user-friendly: TIROL informs the user which plugins are required by an mHealth application. These requirements are specified by the application developer. Then, the user can obtain the involved plugins from an online repository in case s/he has not installed the plugin already for another application. The plugins contain connectors to all devices from the respective domain and select the best one, regarding a certain quality attribute. I. e., most of the steps are executed automatically by TIROL without any user interference.

6 Implementation Description

The prototypical implementation of TIROL is based on the **P**rivacy **M**anagement **P**latform (*PMP*) for Android-based smartphones. Although the presented prototype is based on Android, the underlying concepts can be applied to any other application platform. The PMP is an interlayer for application platforms separating applications from the operating system (see Figure 4). Its main purpose is to provide users a fine-grained and context-based mechanism to restrict the applications' access to private data. To that end, it encapsulates all the sensors of the smartphone into so-called *Resource Groups* and applications have to access these groups via predefined interfaces. As the PMP is extendable, i. e., new Resource Groups can added at any time, it is a suitable foundation for TIROL. Thus, TIROL can be realized as an extension for the PMP and each plugin of TIROL can be mapped to a Resource Group of the PMP. In the following, we focus on the PMP's characteristics which are relevant for TIROL, only. For more information about the PMP, please refer to the respective literature [20, 21].

Since the PMP enables users to limit or prohibit an application's permissions to access a Resource Group at runtime, it is designed to deal with failed access

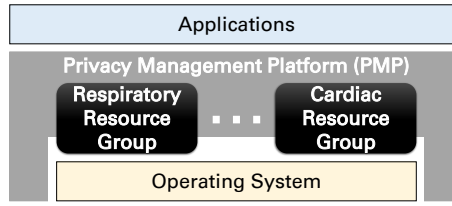


Figure 4: Simplified Layer Architecture of the Privacy Management Platform

attempts and missing data due to such restrictions. For that purpose, the PMP segments the program logic of applications into so-called *Service Features*. The data requirements are assigned to these program sections. Each Service Feature can be deactivated individually in case of missing permissions, i. e., missing data. This is highly advantageous for TIROL. In the mHealth context, a patient might have a subset of the medical devices required for a certain applications, only. As a missing medical device is tantamount to a total restraint of the access to this device, a PMP-based implementation of TIROL is able to deactivate the respective Service Features. Thus, the patient can still make use of the remaining features of the application.

From an application's point of view, a Resource Group represents an interface to a certain kind of data (e. g., location data). The interface specifies generic access functions that abstract from the underlying data sources (e. g., `getLatitude`). The Resource Groups are no inherent parts of the PMP but act as independent services. That way, further Resource Groups can be added to the PMP at any time via an external repository. Accordingly, Resource Groups act similar to the plugins of TIROL.

Each Resource Group bundles several *Resources*. A Resource represents an access method to a certain data source (e. g., GPS). As a consequence, a Resource Group can comprise different implementations for the access functions specified therein. The Resource Groups dynamically select the best Resource[5]—and accordingly the best data source—in the current context. Therefore the connection interfaces of the TIROL plugins can be implemented as Resources. Additionally, Resources can implement a buffer, e. g., to cache the latest sensor data. So the Resource is able to provide these data even if the sensor is currently not connected or available. They are also able to perform domain-specific validation procedures to verify the received data and intervene as necessary.

For each Resource Group so-called *Privacy Settings* are defined. A Privacy Setting is an adjustment method for the Resource Group. The PMP uses the Privacy Settings to restrain the access to the data sources which are managed by the Resource Group. E. g., one of the Privacy Setting for the Location Data Resource Group specifies the maximum permissible accuracy of the location data. While the PMP uses the Privacy Settings as a mechanism to assure the user's privacy by defining an upper limit for the quality of the available data, TIROL

[5] The quality of a Resource is defined respecting a given criterion, e. g., *accuracy*.

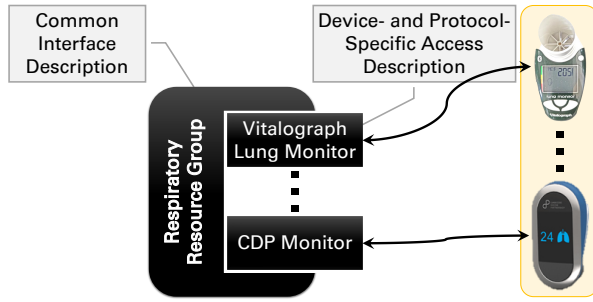


Figure 5: Model of a Health Device Resource Group in the Respiratory Domain

benefits from these settings in another way. If an application does not require the best data quality, this can be expressed via a Privacy Setting. Then the PMP is able to select another less-accurate but energy-saving Resource within the Resource Group in order to reduce the battery drain.

A *Policy Rule* defines which Service Feature requires access to which type of data, i. e., to which Resource Group, and which Privacy Settings are applied to this access. TIROL is able to create such rules optimizing both, data quality and energy performance for any mHealth application and medical devices in this manner. The Policy Rule can be provided via so-called *Presets*.

For the implementation of TIROL, we introduce a model of a generic *Health Device Resource Group* as a blueprint for the plugins of TIROL. Figure 5 depicts the Respiratory Resource Group which is deduced from this generic model. Analogously, there is a Resource Group for each category of medical devices. Within the Resource Groups is one Resource for each medical devices of the respective category. While the Resource Groups define abstract data access functions, the Resources provide the specific implementations of these functions, including the connection establishment and the compliance with the communication protocols.

7 Assessment

In the following, we assess how well TIROL meets the requirements towards a unifying interlayer for mHealth applications (see Sect. 4). Table 1 shows, which component or feature of TIROL fulfills the particular requirement. We consider both, the underlying concept as well as the PMP-based implementation.

The **heterogeneous connectivity** (A) is realized by the connection interfaces of the TIROL plugins. These interfaces are able to bind any medical device to the respective plugin. In the implementation, the Resources establish the connection to the individual devices. The **flexible connectivity** (B) is considered by the TIROL plugins that bundle any medical device providing a certain kind of health data. These plugins are mapped to Resource Groups in the implementation. The "Plug and Play"-like sockets for the plugins provide the **extensible interface** (C) as additional plugins can be added if necessary. The Resource repository of

Table 1: Realization of the Requirements in TIROL's Concept and Implementation

Requirement	Concept	Implementation
(*A*)	Connection Interfaces	Resources
(*B*)	TIROL Plugins	Resource Groups
(*C*)	"Plug and Play"-like Sockets	Resource Repository
(*D*)	Domain-specific Plugins	Domain-specific Resource Groups
(*E*)	Minimal Number Connectors	Economic Data Source Selection
(*F*)	Data Preprocessing in Plugins	Data Preprocessing in Resources
(*G*)	Extensible Plugins	Extensible Resources

the PMP serve the same purpose. The **abstract interface** (*D*) is given by the plugins since each plugin provides only access to data of a certain domain without disclosing from which device this data originates. For the implementation the same holds true for the Resource Groups. The **resource-efficient data access** (*E*) is given by the minimal number of required connectors in the concept. On top of that, the implementation is even able to reduce the power consumption by selecting the most economical data source currently available. Both, the TIROL plugins as well as the Resources are able to preprocess the sensor data. In this way, the data can be validated in order to assure a **reliable data access** (*F*). TIROL and the PMP are compatible to any given device as both can be extended by plugins or Resources. So there are **no dependencies** (*G*) whatsoever.

Therefore both, TIROL's concept as well as its implementation meet all of the requirements towards a unifying interlayer for mHealth applications.

8 Conclusion and Outlook

In times of rising medical costs, novel methods of treatment are required. Especially for chronic diseases, the usage of mHealth applications seems to be highly beneficial. However, users are often repelled by device incompatibilities. Therefore, we postulate a requirements specification for an interconnectivity layer to solve these incompatibility issues. Based on this requirements specification, we devise TIROL, an extensible interconnectivity layer for mHealth applications, and a PMP-based implementation of it. Our assessment shows that TIROL constitutes a sound solution for the incompatibility issues of mHealth applications.

Since privacy concerns are another key issue for mHealth applications [1], our PMP-based implementation of TIROL might also be able to solve this problem. As the PMP's main focus is on privacy and data security [20–22], we could piggyback on these strengths by adding such functionalities to our Health Device Resource Group as well. That way TIROL would be able to provide an mHealth infrastructure assuring privacy and data security.

References

1. Chan, M., Estève, D., Fourniols, J.Y., Escriba, C., Campo, E.: Smart Wearable Systems: Current Status and Future Challenges. Artificial Intelligence in Medicine **56**(3), 137–156 (2012). https://doi.org/10.1016/j.artmed.2012.09.003
2. Continua: H.811 Personal Health Devices Interface Design Guidelines. Specification, Personal Connected Health Alliance (2016)
3. Fang, B., Lane, N.D., Zhang, M., Boran, A., Kawsar, F.: BodyScan: Enabling Radio-based Sensing on Wearable Devices for Contactless Activity and Vital Sign Monitoring. In: Proceedings of the 14th Annual International Conference on Mobile Systems, Applications, and Services. MobiSys '16 (2016). https://doi.org/10.1145/2906388.2906411
4. Gupta, N.: Inside Bluetooth Low Energy. Artech House Publishers (2013)
5. Hester, J., Peters, T., Yun, T., Peterson, R., Skinner, J., Golla, B., Storer, K., Hearndon, S., Freeman, K., Lord, S., Halter, R., Kotz, D., Sorber, J.: Amulet: An Energy-Efficient, Multi-Application Wearable Platform. In: Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems. SenSys '16 (2016). https://doi.org/10.1145/2994551.2994554
6. Hughes, R.D., et al.: Health Device Profile. Specification, Bluetooth SIG (2012)
7. IEEE: ISO/IEC/IEEE Health informatics–Personal health device communication–Part 20601. IEEE Standard ISO/IEEE 11073-20601:2014 (2014)
8. Kouris, I., Koutsouris, D.: Identifying Risky Environments for COPD Patients Using Smartphones and Internet of Things Objects. International Journal of Computational Intelligence Studies **3**(1), 1–17 (2014). https://doi.org/10.1504/IJCISTUDIES.2014.058642
9. Lorincz, K., Chen, B.r., Challen, G.W., Chowdhury, A.R., Patel, S., Bonato, P., Welsh, M.: Mercury: A Wearable Sensor Network Platform for High-fidelity Motion Analysis. In: Proceedings of the 7th ACM Conference on Embedded Networked Sensor Systems. SenSys '09 (2009). https://doi.org/10.1145/1644038.1644057
10. Mare, S., Sorber, J., Shin, M., Cornelius, C., Kotz, D.: Hide-n-Sense: Preserving Privacy Efficiently in Wireless mHealth. Mobile Networks and Applications **19**(3), 331–344 (2014). https://doi.org/10.1007/s11036-013-0447-x
11. Masaud-Wahaishi, A., Ghenniwa, H.: Privacy Based Information Brokering for Cooperative Distributed e-Health Systems. Journal of Emerging Technologies in Web Intelligence **1**(2), 161–171 (2009). https://doi.org/10.4304/jetwi.1.2.161-171
12. Mishra, S.M.: Wearable Android: Android Wear and Google FIT App Development. Wiley Online Library (2015)
13. Murnane, E.L., Huffaker, D., Kossinets, G.: Mobile Health Apps: Adoption, Adherence, and Abandonment. In: Adjunct Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2015 ACM International Symposium on Wearable Computers. UbiComp/ISWC '15 (2015). https://doi.org/10.1145/2800835.2800943
14. O'Donoghue, J., Herbert, J.: Data Management Within mHealth Environments: Patient Sensors, Mobile Devices, and Databases. Journal of Data and Information Quality **4**(1), 5:1–5:20 (2012). https://doi.org/10.1145/2378016.2378021
15. Otto, C., Milenkovic, A., Sanders, C., Jovano, E.: System Architecture of a Wireless Body Area Sensor Network for Ubiquitous Health Monitoring. Journal of Mobile Multimedia **1**(4), 307–326 (2005)
16. Siewiorek, D.: Generation Smartphone. IEEE Spectrum **49**(9), 54–58 (2012). https://doi.org/10.1109/MSPEC.2012.6281134

17. Silva, B.M.C., Rodrigues, J.J., de la Torre Díez, I., López-Coronado, M., Saleem, K.: Mobile-health: A Review of Current State in 2015. Journal of Biomedical Informatics **56**(C), 265–272 (2015). https://doi.org/10.1016/j.jbi.2015.06.003
18. Soceanu, A., Egner, A., Moldoveanu, F.: Towards Interoperability of eHealth System Networked Components. In: Proceedings of the 2013 19th International Conference on Control Systems and Computer Science. CSCS '13 (2013). https://doi.org/10.1109/CSCS.2013.69
19. Sorber, J., Shin, M., Peterson, R., Cornelius, C., Mare, S., Prasad, A., Marois, Z., Smithayer, E., Kotz, D.: An Amulet for Trustworthy Wearable mHealth. In: Proceedings of the Twelfth Workshop on Mobile Computing Systems & Applications. HotMobile '12 (2012). https://doi.org/10.1145/2162081.2162092
20. Stach, C., Mitschang, B.: Privacy Management for Mobile Platforms – A Review of Concepts and Approaches. In: Proceedings of the 2013 IEEE 14th International Conference on Mobile Data Management. MDM '13 (2013). https://doi.org/10.1109/MDM.2013.45
21. Stach, C., Mitschang, B.: Design and Implementation of the Privacy Management Platform. In: Proceedings of the 2014 IEEE 15th International Conference on Mobile Data Management. MDM '14 (2014). https://doi.org/10.1109/MDM.2014.14
22. Stach, C., Mitschang, B.: The Secure Data Container: An Approach to Harmonize Data Sharing with Information Security. In: Proceedings of the 2016 IEEE 17th International Conference on Mobile Data Management. MDM '16 (2016). https://doi.org/10.1109/MDM.2016.50
23. Stollmann: Terminal I/O Profile. Client implementation guide, Stollmann Entwicklungs- und Vertriebs-GmbH (2015)
24. de Toledo, P., Jiménez-Fernández, S., del Pozo, F., Roca, J., Alonso, A., Hernández, C.: Telemedicine Experience for Chronic Care in COPD. IEEE Transactions on Information Technology in Biomedicine **10**(3), 567–573 (2006). https://doi.org/10.1109/TITB.2005.863877

All links were last followed on June 16, 2017.