M2M Application Service Provision: An Autonomous and Decentralised Approach

Michael Steinheimer[1,3], Ulrich Trick[1], Woldemar Fuhrmann[2], Bogdan Ghita[3], and Gregor Frick[1]
[1] Research Group for Telecommunication Networks, Frankfurt University of Applied Sciences, Frankfurt/M., Germany
[2] Department for Computer Science, University of Applied Sciences Darmstadt, Darmstadt, Germany
[3] Centre for Security, Communications and Network Research, University of Plymouth, Plymouth, UK
Email: {steinheimer, trick, frick}@e-technik.org; woldemar.fuhrmann@h-da.de; bogdan.ghita@plymouth.ac.uk

Abstract —This publication presents a novel concept for autonomous and decentralised M2M application service provision. The functional architecture of the approach is introduced as well as a detailed description of the system structure and process for application creation. Furthermore, this publication describes details about the proposed process for decentralised M2M application service management and formal description of M2M application services as well as independent validation of M2M application service configurations. Finally, the prototype M2M system realising the proposed concept is introduced and several scenarios are evaluated.

Index Terms—M2M application service, P2P, service provision

I. Introduction

Devices are becoming more and more intelligent (i.e. they include complex functionality for monitoring and control). Additionally the devices receive the functionality for communication (this enables remote monitoring and controlling of devices). Because of the increasing number of intelligent devices, which support forming of intelligent environments, many new application fields can be established (e.g. as presented in [1] smart building, electro mobility, smart city, energy optimisation etc.). Especially the end user domain is an application field with powerful potential, however not addressable by external service providers.

Machine-to-Machine Communications (M2M) systems realise the integration of such intelligent devices by provision of specific M2M applications. Traditional M2M systems and the corresponding M2M application services have the focus to support specific business processes. Traditional M2M applications provide their functionality as a service for users (end users or companies). Currently existing concepts from standardisation and research to implement M2M application services provision focus on central approaches. Projects from research area are e.g. INOX Managed Service Platform [2], M2M Platform Project based on SOA [3], BOSP Business Operation Support Platform [4], IMS enabled M2M Service Platform [5], e-DSON [6]. Commercial M2M platforms implement acc. [7] mostly the oneM2M standard for M2M [8]. Disadvantages of the centralised approaches are e.g. that the user of the application is dependent of the platform provider as a central instance. Often M2M platforms provide their functionality only for specific application fields and are therefore less flexible. Centralised provision of platforms requires many resources (costs for operation, maintenance, availability, application development). End users often have no possibility to define applications themselves (since the creation of applications requires expert knowledge or the end user does not have access to the platform) or the devices present in user's personal environment cannot be connected to the M2M platforms.

The personal environment of the end user described above is difficult to address by external service providers, since the activities (controlling and monitoring of M2M devices) carried out in this area would severely affect the end user or challenge data security. To antagonise this issue, it was proposed in [9] to integrate the end user. For this purpose, an approach was presented in [9]-[12], which allows end user to define services and automation tasks for their personal environment. The Service Management Framework (SMF) presented in [9]-[11], [13] consists of a local Service Creation Environment (SCE) and a Service Delivery Platform (SDP). The presented SMF integrates the devices present in the personal area of the end users regardless of their technology and provides basic services, e.g. for multimedia communication. The end user can combine intelligent and communicable devices as well as the basic services using the SCE and thus define personalised services for his personal area (e.g. automation tasks, sensor-dependent signalling or monitoring). An elementary requirement for the platform described above was that it provides an opportunity to develop services intuitively. An intuitive development can be realised by a graphical development process and by modelling the behaviour of a system independently from underlying technology. Both were implemented in the above-mentioned approach. The graphical development process and the underlying methodology are discussed in section III.

By the approach proposed in [1], [9]-[13], the end user has thus been given the opportunity to create graphically and intuitively services for his personal environment. An

Manuscript received June 20, 2017; revised September 25, 2017.
doi:10.12720/jcm.12.9.489-499

approach has been defined in [1], [10], and [12] as to how these services can be made available to other end users or organisations in order to gain added value through the use of the services. In [1], [10], and [12] it was proposed that end users combine the individually offered services to provide a more complex, intelligent and more powerful service. As shown in Fig. 1, a complex service consists of the combination of several distributed services that are networked together. Two ways exist to combine services into an application. Multiple end users can offer the same service with the same local functionality running parallel/ synchronously (service aggregation). Alternatively, end users can offer services that have a different functionality and are composed to an application by linking the services together (service composition).

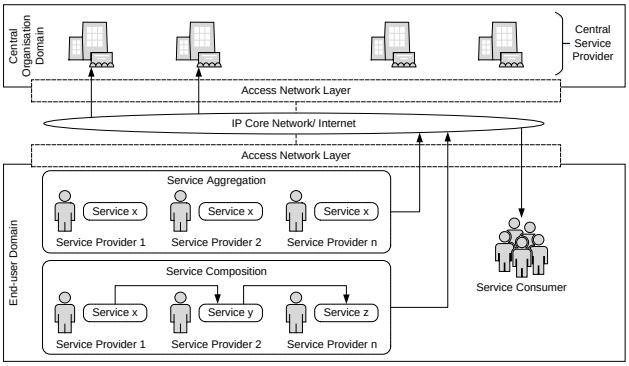

Fig. 1. Cooperative M2M application service provision.

As described above, the central approaches for defining M2M applications have various disadvantages and are not very flexible. Therefore, as a prerequisite for the approach introduced above, it was defined that creation and provision of individual services and the cooperative provision of an application (consisting of distributed services) is exclusively decentralised, i.e. without the integration of logical as well as physical central entities into the overall system. Under these circumstances, the realisation of the approach cannot be implemented with the existing architectures for M2M platforms. Therefore, an approach has been presented in [1], [10], and [12], which allows implementation following a purely decentralised approach.

This publication dedicates to present a more flexible methodology for realisation of M2M systems with the focus on dissolving the bindings to centralised entities, integration of the end user for realisation of M2M applications satisfying the individual requirements of end users, and realisation without specialised and dedicated M2M infrastructures. Furthermore, this publication aims to introduce details about the functional architecture of the proposed concept as well as methodologies forming the basis for the presented concept of autonomous decentralised M2M application service provision.

II. Functional Architecture

In order to clarify how an application in M2M context is structured, definitions of [14]-[16] are used to classify and separate service and application in context of application/ service provision in M2M (illustrated in Fig. 2). It illustrates that an application consists of one or more underlying services that are combined (i.e. aggregated or composed) and, if required, exchange information. An application service contains the application and an application interface to make the application consumable for other entities. The services are realised by one or more service components, which form the building blocks of services. The service components itself are blocks of services. The service components itself are realised via several software applications executed on several execution environments.A service as well as an application can be realised on technical or non-technical principles (i.e. it can be provided using technical devices, e.g. computers, or by a human, e.g. personal assistance services). To distinguish the kinds of services and applications, if required they are indicated as technical service/ application or non-technical service/ application. General indication of both is service/ application.

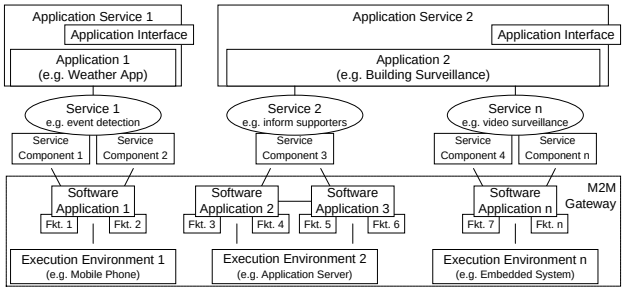

Fig. 2. Classification of service and application.

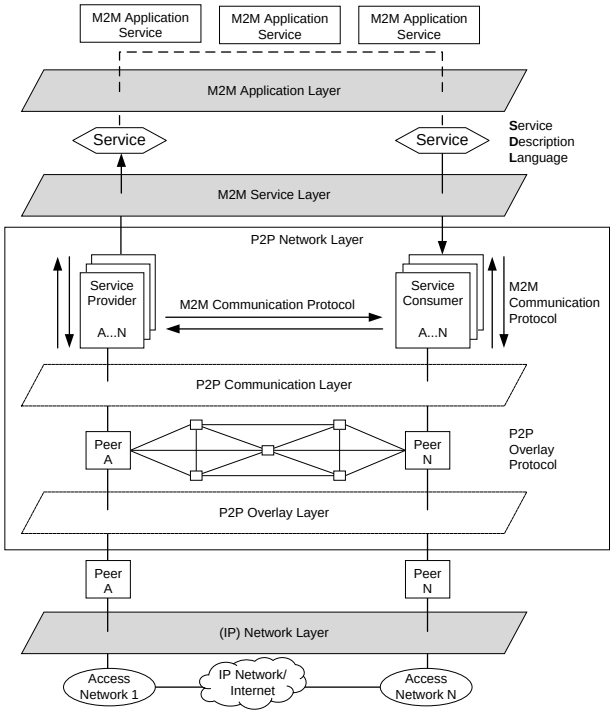

Fig. 3. Functional architecture.

Fig. 3 shows the functional architecture of the concept for the autonomous P2P-based provision of M2M

application services. The architecture is structured into four layers. The following describes the respective layers in their functionality. **Network Layer:** End user's M2M gateways are connected with each another via an IP network, respectively the Internet. End users can be located at different geographic locations as well as in different access networks (wireless mobile networks or fixed access networks), which are interconnected via a core network. **P2P Network Layer:** End user nodes represent equivalent instances and are connected to each other via a P2P network. The P2P Network Layer includes the sub-layer P2P Overlay Layer and P2P Communication Layer. Any data storage in the presented system is decentralised via the end user clients within the P2P Overlay layer. For this purpose, the clients, respectively the peers form a P2P Overlay network. The Overlay network is either a pure structured Overlay network or a pure unstructured Overlay network (due to the general restriction to avoid also partly centralised system architectures). The Overlay network is used as explained in [10], [11], [13] to store the addressing information of the peers (assignment temporary to static contact address). In addition, the P2P Overlay network is used to manage and configure services and applications (described in section IV). The information exchange between the peers for the service utilisation (service requests, confirmations) as well as the necessary signalling to generate the application automatically is also done directly between the peers. For this purpose, various M2M communication protocols (e.g., CoAP, HTTP, SIP, MQTT) can be used. **M2M Service Layer:** The services defined and provided by the end users are available via the M2M service layer and can be used via the corresponding interface. As described in section III and [9], the services are formally described by a service description language. The service interface is described by an interface specification and is available in the M2M Service Layer (explained in section III). **M2M Application Layer:** The distributed M2M application services are implemented within the M2M application layer. For this purpose, the services available via the M2M Service Layer are combined with each other as described in section III. The exchange of information between the services, respectively applications takes place via the P2P network layer by P2P communication protocols. [17]

III. System Structure and Application Creation

In existing platforms for providing M2M applications, the configuration of automation tasks or the creation of application logic is highly dependent on the implementing systems. To prevent this, the concept defined in this research work was designed with a model-based system structure, derived from [18]. Models, expressed by modelling languages, describe model-based infrastructures and systems. The behaviour of an application or a system is described by means of a platform-independent model, separated from the technology-specific realisation of the system [18]. The behaviour of a system can be modelled with a behavioural model in an abstract manner with much lesser complexity than the platform that implements the behaviour of the system or the application [19]. By stepwise abstraction, respectively transformation, an abstract modelled system is transformed acc. [20] into a concrete system by means of multi-step transformation. Fig. 4 shows the transformation level defined for the approach described in this publication. Derived from [20], transformation and mapping rules have been defined to automatically process the transformation from the abstract model of the system created by the user to the platform-specific models shown in Fig. 4 of the novel architecture for provision of distributed M2M applications described in this publication. As described above, the intuitive development of services and applications is done by graphically modelling the behaviour of a service or creating a behaviour model of service/ application. This allows end users to describe the system without having to have any specific knowledge about the execution platform. The behaviour of a system can be modelled intuitively with a state machine. For this purpose, states represent the devices/ services that should be combined, and transitions represent the connections (information flow) between the devices/ services. To enable end user to model a system using state machine concept, a domain independent modelling language is required which can be used to describe the behaviour of a system abstracted in form of a state machine. [17] The following requirements were initially defined for the selection of an optimal modelling language for this purpose: **Machine readability:** Since no further entities are to be involved in the application generation or its implementation, i.e. the application, respectively the service should be generated automatically after modelling, the modelling language must be machine-readable. It must therefore be a formal language. **Standardised language:** It must be a standardised modelling language in order to ensure the easiest portability. **Immediate mapping of graphical and machine-readable notation:** Since graphical modelling by the end user was defined as a central aspect of the concept, a direct mapping of the graphical notation (elements of the state machine) to the machine- readable notation must exist. **Intuitive usability:** Since the system is to be used by an average technically experienced end user after a short training, the complexity of the graphical notation has to be low. Complex and non-intuitive modelling forms reduce or eliminate end user usability. **Existing parser/ interpreter implementation:** To enable further processing the formal language by a machine, an implementation of a corresponding parser or interpreter must exist. **Nested states:** Combination of application services. A system (M2M application service) should be combined with other systems. A previously modelled system is to be reused by being embedded in another

application. **Parallel flows:** Within an application, it should be possible to define parallel sequences in order to implement concurrent tasks. **State parametrisation:** The services that are to be combined must have the possibility to be parameterised (definition of configuration/ input and output parameters). Therefore, the states in the state machine must also be parameterisable. **Conditional state transitions:** The transition of a state or transmission of information to a service or the activation of a service should be provided with a condition. **Synchronisation of states:** To synchronise parallel flows, there must be a possibility for synchronisation. According to these requirements, the following state machine-based modelling languages and modelling concepts have been examined: Business Process Modeling Notation (BPMN), Business Process Modeling Language (BPEL), UML State Machines, Petri Nets, and Finite State Machines. The results of the evaluation are shown in Table I. Requirements the modelling languages/ concepts satisfy are marked with "+"; requirements not satisfied are marked with "-"; requirements partially satisfied are marked with "o". The evaluation of the above-mentioned modelling languages showed that UML State Machines fully meet the requirements. UML State Machines are based on Harel Statecharts, which, due to their structure, meet the structural requirements for the state machine. The standardised description language StateChart XML (SCXML) makes it possible to express Harel statecharts in a formal notation. An SCXML parser, which is also available, enables the machine readability and thus the automatic processing of the formally described state machine. [17]

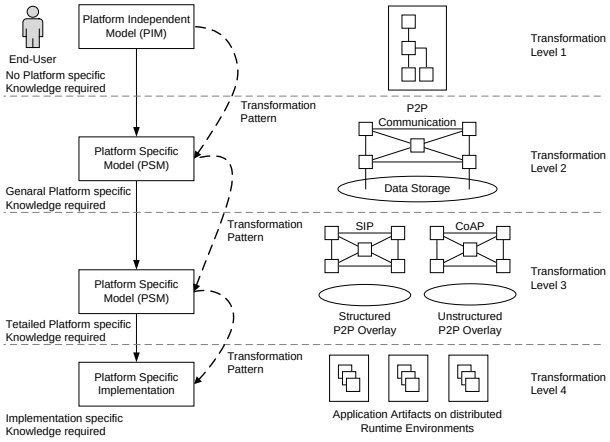

Fig. 4. Transformation level

TABLE I: EVALUATION OF STATE MACHINE MODELLING LANGUAGES

	Modelling Language				
	BPMN	***BPEL***	***UML***	***Petri Nets***	***FSM***
Machine read.	-	+	+	+	-
Standardised	+	+	+	+	-
Immediate map.	-	+	+	+	-
Intuitive usability	+	o	+	-	+
Parser/ Interpreter	-	+	+	+	-
Nested states	o	o	+	+	-
Parallel Flows	+	+	+	+	-
State param.	-	+	+	-	-
Conditional trans.	+	+	+	o	-
Sync. states	o	+	+	+	-

For modelling of a system, the SMF, as described in [9]-[11], [13], provides a GUI to the end user. Using this GUI, the end user graphically creates a state machine that represents the behaviour of the system. The GUI provides an overview of all available devices and services. The services and smart devices that are in the personal environment of the end users, e.g. in its Smart Home, are stored within the SMF in a local repository. The services offered by other peers are stored in the P2P Overlay as described in following section IV. An overview of the services stored in the P2P Overlay is also loaded by the SCE as described in section IV and displayed in the GUI. The end user now graphically combines the available services and devices by first selecting the desired devices or services and then connecting the input and output interfaces defined for the services and devices (transition of the state machine). As described above, the user models the behaviour of service/ application by defining a state machine. The following is a description of the elementary elements of the state machine, as well as the semantic defined by mapping the elements of the state machine to the service/ application components. In addition to these elements, further elements of a state machine can be used, which are not further explained at this point. **State element:** Devices, services, and applications are represented by states. **State parameter:** The state parameters represent the data model of the devices, services and applications. Input, output and configuration parameters can be defined. **Global state machine parameter:** This defines global attributes of a service/ application. Global attributes are available for all services and applications and are queried directly from

the source. **Transition:** The transition from one state to another state (transition target) represents the connection of devices, services and applications and the associated information exchange. **Transition condition:** Transitions can be provided with conditions. This defines the precondition when device, service or application sends an information. Conditions can be optionally defined. **Transition assign element:** The transitions of a state machine have assign elements. These assign elements define how the output parameters are assigned to the input parameters of a device, service or application. Thus, the contents of the messages that are exchanged are specified via this element. **Nested state element:** Using the nested state element enables embedding an already defined state machine, i.e. a service or an application, into another service or application. For this purpose, the corresponding state machine is embedded into a nested state element. The nested state element then is integrated as a simple state into the sequence of the state machine. **Parallel state element:** The parallel state element is used to model parallel sequences that can be synchronised as required at the end. [17]

The interface description of a device, service or application represents the template for a state. It is embedded in a service or application description or managed separately. The interface description specifies the input/ output and configuration parameters as well as the value ranges of a service/ application. The interface description is used as a template to represent the devices, services or applications graphically and functionally in the GUI. In addition to the above parameters, the interface description contains further parameters for classifying and describing a device, service or application by e.g. keywords or prose. The interface description can be stored in a local repository or stored in the Overlay network. In addition, it is possible to request the interface description directly from a peer that offers a service. [17]

The SCE automatically generates the formal description of the end user-modelled system using SCXML. Service/ application are completely described by the generated SCXML document. If it is a local service, the application description is stored locally, parsed, interpreted and executed according to the defined conditions. If it is a distributed application, i.e. consisting of distributed services offered by different peers, the application description is distributed to the peers that offer a service in the application context. In this case, each peer receives only the part of the application description that is required to configure and execute its individual service. Peers therefore do not have an overview of the overall system, which performs as a basis for securing the system. [17]

IV. Application Registry, Configuration, Execution

As described in section II, data storage in the described approach is performed via structured or unstructured P2P Overlay networks. The following is a description of the functions for registering and searching, which are implemented using the Overlay network for the management of services and applications. Services can be offered by different peers. The peers register their services via the Overlay network. To do so, they store the combination of service ID and their associated personal temporary contact information in the Overlay. If different peers offer the same services, several associated peers are stored under the same entry in the Overlay network. Each peer represents a specific instance of a service. Peers can request the Overlay network to get an overview of all available services or a certain subset of all services. This overview is required to generate an application from the various services as described in section III. Sub-sets of services may include e.g. that services are only available for a particular region, have further meta information that can specify a service more detailed, can only be consumed by a particular group, or implement a particular interface description. Depending on the type of request, the Overlay networks offer various ways to provide the overview of the services. For example, the meta-informations stored for a service can be searched by keywords. In order to obtain a specific instance of a service during the application configuration (see below), a peer makes a request (possibly restricted by search criteria) to the Overlay network. This returns a list of all found instances of a service. [17]

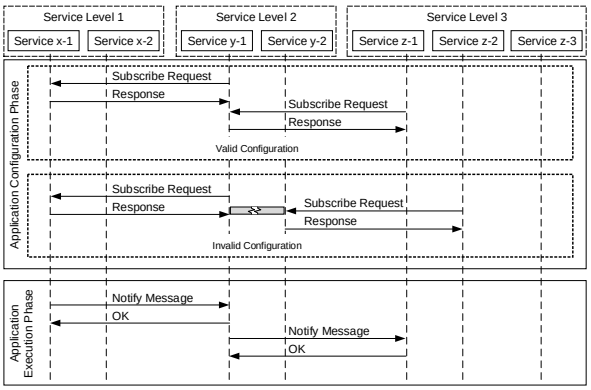

Fig. 5. Application configuration and execution phase.

An M2M application service consists of several services that are interconnected and exchange information. An M2M application is thus a concatenation of distributed services. Each service has input and output interfaces. To realise an application, the output interfaces of a service are connected to the input interfaces of another service, and the information is passed on from one service to the other. The description of the application, i.e., which service is associated with which other service, is defined by a formal application description (described in section III). The approach described in this publication has two phases to realise an application (illustrated in Fig. 5). A configuration phase in which the application is generated automatically and an execution phase in which the application is running.

First, the following section describes the configuration phase. Each peer providing a service involved in an application receives the formal description of the application. As described in section III, each peer only receives the segment of the application description, which contains the necessary information to configure its individual service and insert it into the overall context of the application according to the application description. Inserting into the overall context of the application means that the peer connects the interfaces (input, output) of his service to the other associated services in order to exchange information. By automatically parsing the application description, the peer determines the following information. **Service connections:** To which other services his offered service should be connected. A distinction is made here between input interfaces served by another service and output interfaces which are served by the service itself. **Input interface information content:** Which information is received via the input interface. **Output interface information content:** What information via the output interface should be transferred to another service. **Output interface condition:** Under which prerequisites an information is to be sent to a service via an output interface (directly on request or time-delayed, if a defined condition is met). **Service configuration parameter:** If configuration parameters have been defined for the offered service, the service must be configured accordingly. **Global attributes:** Information about the attributes provided by services that are not directly connected to their own service via an interface. [17]

Each peer now uses the configuration information, determined by the automated parsing of the formal application description, to integrate its service autonomously into the overall context of the application. First, each peer determines the specific instances of the services to be connected to it, i.e. determines a list of peers that offer the service based on the service ID. This is done as described above. The exchange of information takes place in the approach described in this publication according to the Subscribe/ Notify principle. The connection of an input interface of a service with the output interface of another service is performed by sending the peer with its output interface a subscribe request. If the requested service is available, the requested peer confirms service utilisation request with a positive response. If the service is not available or is available later, the requested peer responds with a negative response and rejects to consume the service or defines a later usage time. Once the connections between the instances of the services are established, the local configuration of the service is performed. This consists of the assignment of the defined input and output information content to the interfaces (transition assign). Furthermore, the transfer of any existing service configuration parameters to the local service is performed. In addition, the conditions are configured locally that define the preconditions to transmit an information to a connected service via the output interface (transition condition). [17]

As described above and shown in Fig. 5, the described approach also includes the execution phase additionally to the configuration phase of an application. In the execution phase, the exchange of information takes place between the peers, which represents the M2M application service. Once the connections between the service instances have been established, the information is transmitted between the services using Notify messages. Sending a notification message either is done directly on request (global attribute or condition less transition) or as soon as a defined condition (transition condition) applies. [17]

It should be noted that the same service can be provided by several instances (peers). This may result in incomplete configurations of an application during the configuration phase. In addition to a complete configuration of an application, Fig. 5 also shows an incomplete configuration of an application). For this purpose section V presents an algorithm allowing all peers to determine independently if they have embedded themselves in a valid application configuration. The algorithm developed for this purpose is based on the computation of the transitive closure derived from [21] of all connections established in the P2P network in the application context. If a peer detects that it has embedded itself in an invalid application configuration, the established connections are automatically disconnected and, if necessary, a new local configuration process is performed. Thus, it is possible to determine independently a valid application configuration without having to have a complete view of the system. [17]

V. Application Validation

Fig. 5 shows in the Invalid Configuration section that no path exists between the first services (service at most left service level) and the last services (service at most right service level). This means, although the peers have correctly configured their individual service connections according the application description, the application was not configured correctly in total. A correctly configured application contains at least one closed path through all service levels, starting at an instance at first service level and ending at an instance of a service at the most right service level. Because the invalid configurations are unusable for the application configuration, they first must be identified as described in the following and afterwards removed again.

The peers require an overview of the overall configuration of the application in order to identify independently whether they have established an invalid connection to another peer. I.e. they need an overview of peers included in a specific instance of the application service connected to other peers. In order to provide such an overview of the overall configuration, the peers must document to which other peers they have established a

connection. This documentation have to be available to all other peers, which in turn use the documentation to identify the invalid connections. As the P2P Overlay inside the P2P network layer is a shared database, it is used to store the connection information.

The connections of the services are mapped to a directed graph by representing the peers as nodes and the connections between the peers as edges between the nodes. Arithmetic operations applied to that graph enable to determine whether a closed connection exists from an initial node to an end node. All nodes whose connections are not in the path between the start and end nodes have invalid connections and must remove them again. The structure of the graph, represented as adjacency matrix, is stored as a data structure inside the Overlay. The structure of the adjacency matrix is defined as follows. The nodes of a graph are specified as row and column names in the adjacency matrix, and the connections between the nodes as entries in the associated fields in the adjacency matrix. Because it is a directed graph, the nodes from which a connection originates are specified in the row names, and the destination nodes of these links in the column entries. A uniform name for the column and row entries is defined as starting with the service ID as prefix and appending the specific peer ID as a suffix.

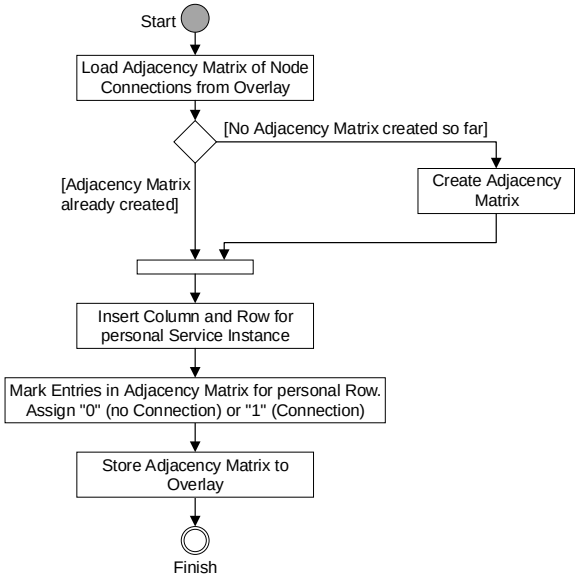

Fig. 6. Algorithm for documentation of connection establishments.

If a peer has established a connection according to the application description with another peer, respectively has associated a service offered by him with another service, it executes the algorithm shown in Fig. 6 in order to document the established connection in the adjacency matrix. The following describes how peers can use the connection overview stored in the Overlay in order to identify and remove the invalid connections.

The arithmetic operations and algorithms applicable on graphs allow the evaluation of characteristics of a graph. The Transitive Closure of a binary relation in a graph (link between nodes), calculated e.g. by the algorithm of Warshall [21], indicates which node pairs are mutually accessible. This information determines whether a destination node is reachable from a start node, possibly with the inclusion of other nodes. All nodes contained in the transitive closure of a node can be accessed from the associated node. Since the connection graph is stored in the Overlay, all peers request and use it to compute the transitive closure of the connection graph independently. Thus, each peer gets an overview of which peers are reachable among each other. In order to verify the complete configuration of an application, each peer, for its individual node in the connection graph autonomously executes the algorithm described in Fig. 7. First, the peer queries the complete adjacency matrix of the connection graph from the Overlay. For the resulting connection graph, the peer then calculates the transitive closure. Thus, the peer has an overview of which nodes are mutually accessible. The column entries for the nodes on the last service level are determined afterwards. The determination of the services at the last service level is made possible since the last services in the formal application description are marked as "final state". This determines the prefix of the column description. The calculating peer does not know the suffix, but by the uniformly defined format of the column names enables the determination of the column name without knowledge about the suffix. After identifying the nodes at the last service level, it is checked whether these nodes, starting from the own node, are reachable. If there is not at least one instance of each service at the last service level reachable, an invalid configuration of the application exists and the connection to the peers must be removed again. If a connection to the instances at the last service level exists, the next step checks whether the own node is accessible by an instance of a service on the first service level. For this purpose, the row ID of the nodes on the

first service level is determined first. The determination of the service on the first service level is made possible by marking the first service in the formal application description as "initial state". This determines the prefix of the line description. Similar to the determination of the column ID, this makes it possible to identify the row IDs of nodes on the first service level. After identifying the nodes on the first service level, it is checked whether the own node is accessible from one of these nodes. If this is the case, a continuous connection from the first and the last service level exists.

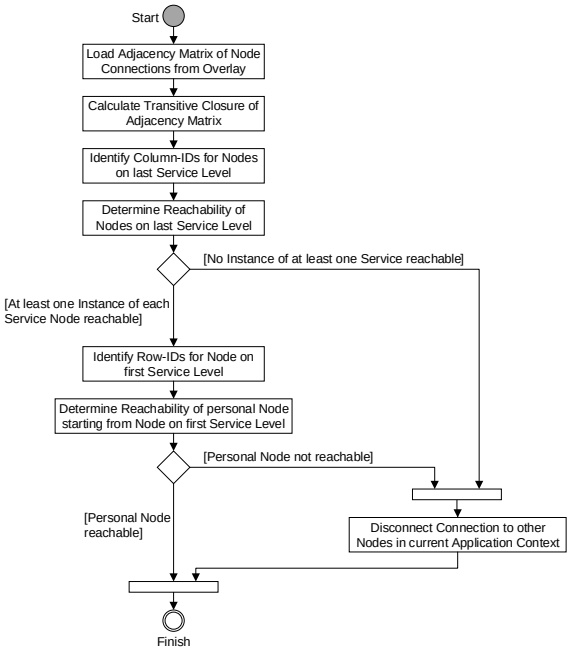

Fig. 7. Algorithm for determination of invalid application service configuration.

Incomplete configuration of the application exists when: first, at least one instance of each service on the last service level of the connection graph cannot be reached from its own node and/ or second, the own node cannot be reached by at least one instance of the service at the first service level of the connection graph. If a peer detects that it is within an invalid configuration of the application, it removes the connections to other peers. Removing a connection between the peers is done by sending a request to the connected peer with the request to disconnect.

VI. Prototype and Evaluation Results

In order to verify the above introduced concepts for autonomous decentralised M2M Application Service Provision as well as to evaluate the behaviour of the P2P Network Layer a prototype has been realised implementing the described M2M service platform. This section presents the testbed setup and the performance results. As prototype hardware a high performance server system is used equipped with 2x24 CPUs und 2x64 GB RAM. This server system contains the hypervisor OS Proxmox [22] that provides virtual host systems and virtual networks. This makes it possible to implement the prototype system without affecting the measurements by activities that can occur in a physical network. In order to implement the scaling of the M2M system, the container virtualisation system Docker [23] is used within the virtual hosts. This makes it possible to emulate a large number of standalone containers, i.e. peers, without being restricted by the hardware requirements of virtual machines, as would be the case when using a hypervisor virtualisation solution. Fig. 8 shows the prototype setup. It consists of two VMs each implementing a Docker host. Docker hosts are connected to each other via two separate virtual networks. The Docker hosts contain several Docker containers whose number is scalable as well as internal networks. Through the internal networks, different access networks are emulated via which the peers, represented by the Docker containers, are connected. This makes it possible to evaluate not only the P2P system itself, but also the behaviour of distributed M2M applications in different network topologies. The prototype includes two separate networks to which the Docker hosts as well as the peers themselves are connected: the P2P/ M2M Network (illustrated with solid lines) and a separated Management Network (illustrated with dashed lines). The management network is used to control the individual peers, perform and control performance measurements as well as to log them without affecting the behaviour of the networks over which the peers are connected. Two additional VMs are also connected to the management network: the Control Logic Server, which contains the control logic of the prototype, and a MySQL server that is used to log measurement results. The Docker containers contain the distributed application logic to provide the M2M services and to perform the autonomous configuration of the M2M application. Based on this prototype setup the behaviour of the P2P network layer was analysed. The M2M communication protocols SIP and CoAP has been analysed as well as unstructured P2P Overlay networks represented by the Gnutella architecture [24] and structured P2P Overlay networks represented by the Chord architecture [25]. The analysis results are explained below. Table II shows the comparison of the M2M communication protocols SIP and CoAP for various communication scenarios for M2M application configuration and execution. It can be seen here that the number of packets sent and the required data volumes when using CoAP is significantly lower than with SIP. The use of CoAP offers advantages with regard to the required number of packages as well as the data volume. If an existing telecommunication infrastructure is to be used for peer communication, SIP offers the advantage that no additional protocol stack is required. The network resources when using SIP are significantly reduced if the alternative TinySIP messaging library [26] is used for message exchange. It enables sending SIP messages with a maximum message size of 29 bytes. Fig. 9-12 shows

the analysis of the significant characteristics of P2P Overlays with scaling number of peers. Fig. 9 shows the effort for inserting/ searching/ removing operation of a data record in a structured P2P Overlay, as well as the initial entry of a peer to the Overlay. The effort in scaling the peers does not increase significantly. In comparison, Fig. 10 shows the effort for searching a data record in an unstructured P2P Overlay on the basis of the transmitted messages. The effort increases linearly to the number of peers and is dependent on the number of connections between nodes (ABN). This is caused by the Query Flooding process for message forwarding, which is typical for unstructured P2P systems. In Fig. 10 the queries (Query) and the corresponding responses (Query-Hit) are shown. The queries are significantly higher than the answers. Compared to structured P2P Overlays, this is a crucial disadvantage. Both types of P2P Overlays require a stabilisation process of the network, which is executed continuously parallel to the data record requests. The effort for both approaches scales linearly. However, this is dependent on the respective interval of the stabilisation process of the algorithm. The analysis shows that a structured P2P Overlay requires significantly more effort for the stabilisation process than an unstructured Overlay. This is justified by the fact that the algorithm for validating the topology of the Overlay with a structured P2P Overlay is much more complex. In the case of an unstructured P2P Overlay, the simple request of accessibility of neighbouring nodes is done. The analysis of the P2P Overlay approaches has shown that the effort for finding a data record in an unstructured P2P Overlay is significantly higher than with a structured P2P Overlay. An unstructured P2P Overlay, offers the advantage that the effort to re-organise the Overlay when a peer leaves is significantly less. This is particularly advantageous when a frequent fluctuation of peers exist. Furthermore, in structured P2P Overlays the finding of an existing data record is guaranteed; on the other hand, this aspect cannot be ensured in unstructured P2P Overlays. In unstructured Overlays, however, a keyword-based search is possible.

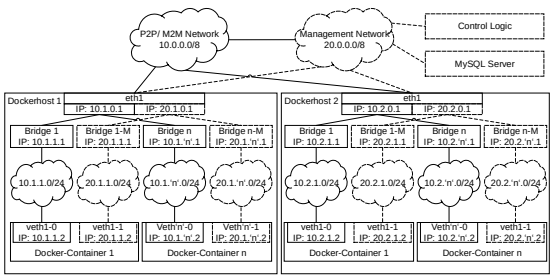

Fig. 8. Prototype setup.

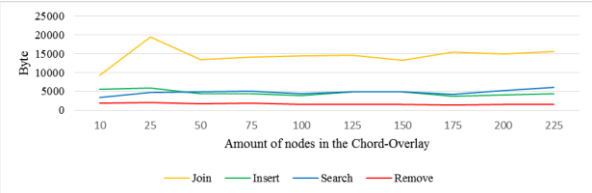

Fig. 9. Operations in structured P2P overlay.

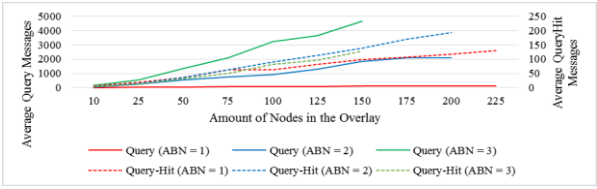

Fig. 10. Dataset request operation in unstructured P2P overlay.

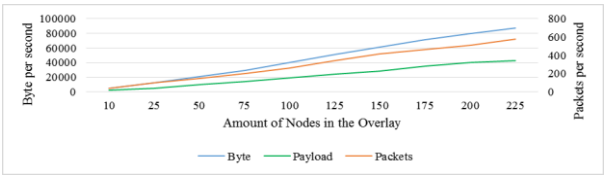

Fig. 11. Stabilisation process of a structured P2P overlay

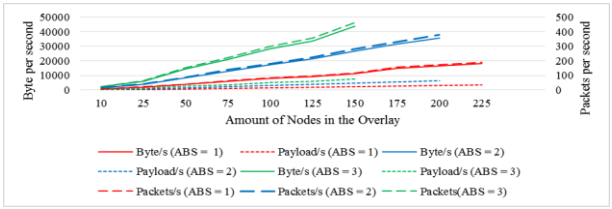

Fig. 12. Stabilisation process of an unstructured P2P overlay.

TABLE II: EVALUATION OF M2M PROTOCOL BEHAVIOUR

Use Case	M2M Protocol			
	CoAP		SIP	
	Packets	*Bytes*	*Packets*	*Bytes*
Successful subscription establishment	2	138	6	2593
Unsuccessful subscription establishment	2	119	2	837
Sending notification	2	126	2	913
Subscription refreshment	2	139	4	1743
Subscription termination by a subscriber	2	126	4	1718
Subscription termination by a notifier	2	120	2	860

VII. CONCLUSIONS

This publication presented principles and details of a novel concept for autonomous and decentralised M2M application service provision. This concept offers new possibilities for applications, realised by several peers, independent of central appliances or corporations. Especially the independent structure of presented system architecture and the simplicity of M2M service/ application creation combined with the autonomous management of M2M application services form the strength of the proposed framework and offer a promising approach for supporting M2M networks with end user environment integration. The analysis of the M2M communication protocols and the P2P Overlay architectures has shown that all of them have their strengths and weaknesses. Optimal selection of the M2M protocols, respectively P2P Overlay architecture depends on the application scenario and network topology.

ACKNOWLEDGMENT

The research project P2P4M2M providing the basis for this publication is partially funded by the Federal Ministry of Education and Research (BMBF) of the Federal Republic of Germany under grant number 03FH022IX5. The authors of this publication are in charge of its content.

REFERENCES

[1] M. Steinheimer, U. Trick, W. Fuhrmann, and B. Ghita, "P2P based service provisioning in M2M networks", *Proc. of Sixth International Conference on Internet Technologies & Applications (ITA 15)*, Wrexham, Wales, UK, September 2015.

[2] S. Clayman and A. Galis, "INOX: A managed service platform for interconnected smart objects," in *Proc. Workshop on Internet of Things and Service Platforms*, December 2011, pp. 1-8.

[3] S. Zhang, J. Zhang, and W. Li, "Design of M2M Platform Based on J2EE and SOA," in *Proc. International Conference on E-Business and E-Government*, September 2010, pp. 2029–2032.

[4] Q. Xiaocong and Z. Jidong, "Study on the structure of 'Internet of Things (IOT)' business operation support platform," in *Proc. 12th IEEE Int. Conf. Commun. Technology*, November 2010, pp. 1068-1071.

[5] L. Foschini, T. Taleb, A. Corradi, and D. Bottazzi, "M2M-based metropolitan platform for IMS-enabled road traffic management in IoT," *IEEE Commun. Mag.*, vol. 49, no. 11, pp. 50-57, November 2011.

[6] Y. J. Kim, E. K. Kim, B. W. Nam, and I. Chong, "Service composition using new DSON platform architecture for M2M service," in *Proc. International Conference on Information Network*, February 2012, pp. 114-119.

[7] J. Kim, J. Lee, J. Kim, and J. Yun, "M2M Service Platforms: Survey Issues and Enabling Technologies," *IEEE Communications Surveys & Tutorials*, vol. 16, no. 1, pp. 61-76, First Quarter 2014.

[8] oneM2M Standardisation Committee. oneM2M Release 2 Specifications. [Online]. Available: http://www.onem2m.org/technical/published-documents

[9] M. Steinheimer, U. Trick, W. Fuhrmann, and B. Ghita, "Load reduction in distribution networks through P2P networked energy-community," in *Proc. Fifth International Conference on Internet Technologies & Applications*, Wrexham, Wales, UK, September 2013.

[10] M. Steinheimer, U. Trick, W. Fuhrmann, and B. Ghita, "P2P-based community concept for M2M Applications," in *Proc. Second International Conference on Future Generation Communication Technologies*, London, UK, December 2013.

[11] M. Steinheimer, U. Trick, P. Ruhrig, R. Tönjes, M. Fischer, and D. Hölker, "SIP-basierte P2P-Vernetzung in einer energie-community," *ITG-Fachbericht 242: Mobilkommunikation*, VDE Verlag GmbH, Osnabrück, Germany, Mar. 2013, p. 64.

[12] M. Steinheimer, U. Trick, W. Fuhrmann, and B. Ghita, "P2P-based M2M community applications," in *Proc. Eleventh International Network Conference*, Frankfurt, Germany, July 2016.

[13] M. Steinheimer, U. Trick, P. Wacht, and P. Ruhrig, "Decentralised optimisation solution for Smart Grids using Smart Market aspects and P2P internetworked Energy-Community," in *Proc. IEC/ SGCC/ VDE World Smart Grid Forum*, Berlin, Germany, September 2013.

[14] ITU-T Y.101 (2000), Recommendation, Global Information Infrastructure Terminology: Terms and Definitions, ITU, 2000.

[15] ISO IEC 20000-1:2011 (2011) Part 1: Service Management System Requirements, IEEE, 2011.

[16] ITIL V3.1.24 (2007) Glossary of Terms, Definitions and Acronyms, ITIL, 2007.

[17] M. Steinheimer, "Optimierte P2P-Dienstearchitektur für hochverfügbare M2M-Applikationen (P2P4M2M)," TR-004, Frankfurt, Germany, October 2016.

[18] Object Management Group. OMG Model Driven Architecture. (January 2017). [Online]. Available: http://www.omg.org/mda/

[19] Object Management Group, "Model Driven Architecture (MDA) MDA Guide rev. 2.0," Boston, USA, June 2014.

[20] R. Petrasch and O. Meimberg, "Model Driven Architecture," dpunkt.verlag, Heidelberg, Germany, 2006.

[21] V. Turau, Algorithmische Graphentheorie De Gruyter Oldenbourg, Munich, Germany, 2009.

[22] Proxmox Server Solutions. [Online]. Available: https://www.proxmox.com/en/, April 2017.

[23] Docker. (April 2017). [Online]. Available: https://www.docker.com/

[24] The Gnutella Protocol Specification v0.4. (April 2017). [Online]. Available: http://web.stanford.edu/class/cs244b/gnutella_protocol_0.4.pdf

[25] I. Stoica, R. Morris, D. Karger, M. Kaashoek, and H. Balakrishnan, "Chord: A scalable peer-to-peer lookup service for internet applications," *IEEE/ACM Transactions on Networking*, vol. 11, 2013.

[26] TinySIP, European Patent Specification EP 1 968 275 B1, October 2016.

Michael Steinheimer received the B.Sc. degree from the University of Applied Sciences Darmstadt, Germany, in 2008 and the M.Sc. degree from the University of Applied Sciences Darmstadt, Germany, in 2011, both in computer science. He is currently pursuing the Ph.D. degree with the Centre for Security, Communications and Network Research, University of Plymouth, UK. His research interests include decentralised service provision, Machine-to-Machine Communications, and Peer-to-Peer Networks.

Ulrich Trick received the Dipl. Ing. degree from the University of Kaiserslautern, Germany, in 1983 in Electrical Engineering - telecommunications. He received his Doctoral degree from the University of Kaiserslautern, Germany, in 1987. Since 2001 he is Professor for Telecommunication Networks with the Department for Computer Science and Engineering at Frankfurt University of Applied Sciences, Germany. His research interests include NGN, M2M, IoT, P2P and virtualisation.

Bogdan Ghita received the Dipl. Eng. from Politehnica University of Bucharest, Romania, in 1998 and his PhD from Plymouth University, UK, in 2005. He is Associate Professor at Plymouth University and leads the networking area within the Centre for Security, Communications, and Network research. His research interests include computer networking and security, focusing on the areas of network performance modelling and optimisation, wireless and mobile networking, and network security. He has been principal investigator in several industry-led, national, and EU research projects. He is the chair of the International Networking Conference series.

Gregor Frick received the B. Eng. degree in Electrical Engineering and Information Technology from the Frankfurt University of Applied Sciences, Germany, in 2014 and the M. Eng. degree in Information Technology from the Frankfurt University of Applied Sciences, Germany, in 2017. He is currently preparing his Ph.D. course with the Centre for Security, Communications and Network Research, University of Plymouth, UK. His research interests include Wireless Mesh Networks, Network Functions Virtualisation, and Disaster Networks.

Woldemar Fuhrmann is Professor at Department for Computer Science at the University of Applied Sciences Darmstadt, Germany. His research interests include Next Generation Mobile Networks and Telecommunications.