Mobile Networks and Applications manuscript No.
(will be inserted by the editor)

CollaborateCom Special Issue
Combining Mobile XMPP Entities and Cloud Services for Collaborative Post-Disaster Management in Hybrid Network Environments

Ronny Klauck · **Michael Kirsche**

Received: date / Accepted: date

Abstract Crises such as the Fukushima incident in Japan showed the demand for flexible and easy-to-use monitoring and communication systems to support post-disaster management (i.e. the organization of actions in the follow-up of disasters), especially when critical infrastructure is affected. Such systems can effectively only be realized with a merging of various device classes and the integration of mobile actors and wireless communication technologies to provide the necessary flexibility. This article introduces a system design that combines portable hand-held devices as well as autonomous sensors through XMPP with the flexibility of cloud services to support post-disaster management. This combination provides the communication between the different involved parties (e.g., rescue teams, relief forces, NGOs) and enables a global view on sensed data through the use of cloud-based storage and analysis services. Along with a discussion about requirements and a description of appropriate solutions and initial evaluations, we present new insights on the practical appliance of XMPP and potential enhancements for XMPP-based real life collaboration applications in hybrid (ad hoc and infrastructure) network scenarios. We also show that resource constrained devices can run the XMPP protocol to extend smartphones with sensors or to connect different device classes in a seamless way.

Keywords Collaboration · XMPP · Cloud Services · mDNS · Post-Disaster Management

R. Klauck · M. Kirsche (✉)
Computer Networks and Communication Systems Group,
Brandenburg University of Technology Cottbus,
Walther-Pauer-Street 2,
03046 Cottbus, Germany
R. Klauck
E-mail: rklauck@informatik.tu-cottbus.de

M. Kirsche (✉)
E-mail: michael.kirsche@tu-cottbus.de

1 Introduction

Over the last decade, a convergence of mobility and computing was observed with a rapid dissemination of portable computers in various forms (embedded vs. stand-alone) and types (smartphones vs. notebooks). The portability of computing technology is nowadays combined with the ubiquitousness of communication technologies, again in various forms (wired vs. wireless) and types (short vs. long range). A combination of these aspects leads to a pervasive access for communication and computing, which expresses itself in a wide array of applications. One particularly interesting use case is post-disaster management, which embraces possibilities as well as challenges from the combination of the pervasive availability of communication and computing at the same time. We introduce a system designed to support data acquisition and representation as well as user collaboration with an emphasis on ubiquitous communication and computing in the area of post-disaster management.

Threats and disasters, no matter if natural or man-made, are unpredictable yet hard to control or avoid. A critical aspect in the follow-up of disasters is a quick and accurate evaluation of the circumstances to support relief forces in the safeguarding of human lives and critical infrastructures. Relief forces are usually guided by disaster management teams, which need an overview of the incident and the current situation. If the management team has no access to status information, correct decisions about instructions are difficult, as the accidents in the Fukushima nuclear power plant following the earthquake and tsunami catastrophe in Japan, have shown. Rescue teams and relief forces are often in doubt about the best way to handle a critical situation when they have no guidance or overview of an incident. It is therefore essential to support post-disaster management teams as well as relief and rescue forces with a quickly deployable and easy-to-use system that supports communication and collaboration of all involved parties as well as easy access to monitored environmental data (e.g., radiation levels for the Fukushima accident). To fulfill this need for a flexible, user-friendly, and co-operative solution, we propose a system of cooperating XMPP-driven entities combined with collaborative cloud services to support data gathering and analysis as well as data access for all involved parties.

Reliability, flexibility, cost-efficiency, and a wide applicability of the system design are our main concerns. Therefore we choose the openly standardized *Extensible Messaging and Presence Protocol* (XMPP) [23] as the basic communication protocol. XMPP is a set of flexible and open XML technologies standardized by the IETF and widely deployed and used in the Internet. Through its decentralized client / server architecture, XMPP is fault tolerant and minimizes the single point of failure problem. Deployed, for example, in applications running on mobile hand-held devices equipped with exchangeable sensors, XMPP allows partners to interact and communicate as well as to collect data from sensors over different IP-based communication technologies and share it with other devices or users. The applications and the underlying technology should enable communication and data sharing inside the local *Peer-to-Peer* (P2P) user group as well as sharing the data with other users over the Internet or saving it in a cloud-based data storage pool [20] for further analysis and processing with cloud services. Various cloud-based services are possible, e.g.: web browser access of sensor data for third parties, simplifying decision processes with ad hoc cooperation of rescue teams, posterior data analysis with statistical processing inside the cloud.

A vital aspect of XMPP is the choice of interacting with a server infrastructure (XMPP Core) as well as alternatively using ad hoc communication (P2P) by using the *XMPP extension* (XEP) *XEP-0174 Serverless Messaging*. In general, XEPs[1] extend XMPP with additional functions. XEP-0174 [22] enables ad hoc communication through the use of *DNS Service Discovery* (DNS-SD), *Multicast DNS* (mDNS) and *Zeroconf* [27]. A major issue is the unique addressability of XMPP entities in both infrastructure and ad hoc mode. Uniqueness of JIDs is currently not provided in ad hoc cases, in contrast to infrastructure scenarios where a central XMPP server provides unique *Jabber IDs* (JIDs). We address this issue by introducing an XMPP-driven data collecting system designed especially for hybrid and ubiquitous network environments, where data needs to be located both in ad hoc as well as infrastructure modes with support for unique JIDs.

This work provides several contributions: We present the system design with an analysis and initial practical evaluation of the underlying networking aspects (e.g., switching between infrastructure and ad hoc mode, providing unique JIDs). With the evaluation, new insights on the practical compliance of XMPP for collaborative applications are gained and presented. We also show that the requirements to assist post-disaster management are met by our XMPP- and sensor-equipped hand-held devices in combination with the flexible cloud-based storage and processing services and with the inclusion of external and autonomous sensor devices. We verify the cooperation of our XMPP-driven hand-held devices with additional autonomous sensor devices through a prototypical implementation and practical measurements on a operating system for resource constrained devices.

The article is organized as follows. Section 2 introduces our scenario and corresponding requirements. The state-of-the-art is surveyed in Section 3 with a discussion of related work in Section 3.3. The system design is introduced in Section 4 together with a discussion of the current implementation state and possible extensions. Section 5 summarizes our initial evaluations and the prototypical verification. Section 6 concludes this work.

2 Scenario and Use Case Considerations

Crisis and post-disaster management is never a fixed nor predetermined work. Because each crisis has its particularities, approaches that might work in a certain place, time, or stage might not work in a different one. It is hence important to gain as much information as possible about the incident and the status of affected people, buildings, landscapes, and critical infrastructures (e.g., communication and transport networks). Only then can rescue and relief forces take the best possible steps according to the current situation.

Nowadays, early warning systems and sometimes even sensor networks are already deployed to cope with natural (e.g., earthquakes, tsunamis, floodwaters, tornadoes, fires) or man-made (e.g., terrorist attacks) threats. Action plans for disaster cases are created and critical infrastructure is well-protected against various kinds of threats. However, recent disasters like the earthquake / tsunami catastrophe in Japan showed that pre-planned precautions are often not sufficient and deployed sensors and communication infrastructures get destroyed, hence becoming unusable by rescue forces. This case is depicted in Figure 1, where flooding destroyed the communication infrastructure (e.g., landline, UMTS) and the status of the critical infrastructure (power plant) is unknown to the rescue forces (depicted by question marks). Such examples require a post-disaster sensor and monitoring system that needs to be flexible and easily deployable to support crisis management and rescue forces in every possible way.

Figure 1 also depicts the complexity of an exemplary post-disaster case. Several stakeholders are typically interacting with each other, ranging from management teams to different rescue forces (e.g., fire department, police forces, technical relief) and third party volunteers. It is important to consider volunteers and victims as well. We learned in

[1] List of XMPP Extensions (XEP) [Online] http://xmpp.org/xmpp-protocols/xmpp-extensions/

interviews with disaster management stakeholders that currently used systems do not consider volunteers, although they might provide additional capabilities and information sources in post-disaster scenarios. Affected humans on the other end need to be informed of immanent danger (e.g., radioactive zones) through a wide array of information channels. It is necessary to consider these specifics and include volunteers into our system design and provide the management team with possibilities to inform victims.

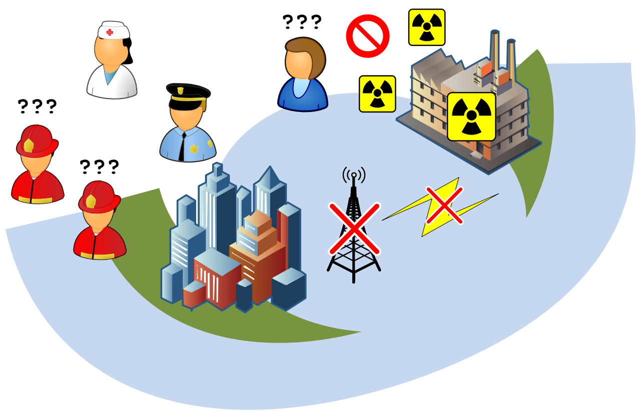

Fig. 1 Exemplary disaster scenario

Our reference scenario comes with certain specifics and issues that need to be considered for any solution, namely:

– Variety of rescue forces in numbers and type involved;
– Sensor devices are typically purpose-specific;
– Normally only local view on self-measured data;
– Proprietary communication solutions deployed;
– No use of open standards, hard to interface with;
– Special training required to operate the system.

Derived from these issues is the following list of goals we want to achieve with our design and solution approach:

– Support different communication parties / partners;
– Integrate volunteers with commodity hardware;
– Provide a global view on self- and remotely-sensed data;
– Allow access for locally and remotely involved parties;
– Provide different information channels;
– Use open and standardized communication protocols;
– Enable data access with widely available applications;
– Provide simple data storage and posterior data analysis capabilities through cloud-driven services;
– System shall be easy-to-use and flexible to deploy.

Our system itself is described elaborately in Section 4. In general, the system is designed for shorter working periods (days to several weeks), in contrast to typical wired or wireless sensor network deployments for disaster alerts or monitoring [18], which are supposed to run for longer time periods (several years). For long range WWAN communication we use standardized technologies like GPRS or UMTS, while IEEE 802.11 WLAN is used to provide ad hoc communication on-site. Short range communication standards like Bluetooth or IEEE 802.15.4 are used to wirelessly attach external and autonomous sensor devices. In general, we favor standards instead of proprietary solutions. Furthermore, no specialized software should be required to access the currently sensed or remotely stored sensor data. Sensor data measured by rescue forces will assist crisis management in various ways, namely in getting accurate incident reports and in making adequate real-time situation-dependent choices. A further combination of sensed data, place and time of the sensing event, and the specific sensing entity (e.g., fire fighter, rescue specialist) enables an even wider range of applications, such as: environmental monitoring, marking of dangerous spots for non-involved bystanders, or traceable scanning of sites for endangered civilians.

3 Technical Background and Related Work

This section summarizes the state-of-the-art of components and protocols that we use in our system. It concludes with a discussion of and separation from related research work.

XMPP provides a collection of open technologies to realize Instant Messaging (IM), user collaboration, and Voice-over-IP (VoIP). The architecture of XMPP is based on a decentralized client / server structure [23]. This means that XMPP clients send messages to their domain specific XMPP server while servers from foreign XMPP domains can also communicate with each other to forward messages. XMPP clients are interconnected by a domain-based network of servers. A single point of failure or overload situations can be prevented because a server malfunction will only affect a separated domain and not the whole XMPP network. The main advantage of XMPP as the underlying communication protocol is that the XMPP Standards Foundation offers an open, standardized, and continuously maintained protocol family, allowing system designers to benefit from the aspects of sustainability. Furthermore, XMPP offers a rich variety of open source software[2] for servers, clients, and libraries supporting several operating systems, thus reducing development costs. A diversity of systems, ranging from desktop computers to mobile entities, can easily be connected through XMPP. It supports various application types beside the usual messaging or presence propagation. Examples can be found in the field of ad hoc grid computing [29] or inter-cloud directory and exchange [2].

Another benefit is that XMPP can be enabled to communicate over LANs or WANs without the need for server infrastructures by using *XEP-0174 Serverless Messaging* [22].

[2] XMPP Software [Online] http://xmpp.org/xmpp-software/

This method uses the zero-configuration networking principle (refer to the following Subsection 3.1) to discover entities that support the protocol and then to negotiate a serverless connection to exchange XMPP messages in the end.

3.1 mDNS & DNS-SD

Multicast DNS (mDNS) [4] is based on the Domain Name System (DNS) protocol [17]. The main purpose of DNS is to map domain names to network addresses. DNS specifies the roles of the service provider (server) and the service user (client). The client sends the request for a given domain name to the server and receives the corresponding network address. In contrast to that, mDNS can resolve domain names without the help of any server. Each node in the local (ad hoc) network stores its own list of DNS resource records (e.g., `A`, `SRV`, `TXT`) and joins a multicast group. When a device in the ad hoc network wants to know the address of another node it sends the request to the multicast group. The node with the corresponding domain name (included in the `A` record) replies with its network address. For mDNS the multicast addresses `224.0.0.251` (IPv4) and `ff02::fb` (IPv6) as well as the UDP port `5353` are likewise reserved by the IANA[3]. DNS Service Discovery (DNS-SD) [3] enables locating and publishing of services inside a network. DNS-SD uses so-called DNS resource records to provide information about services. A node offers his service by propagating the following DNS records:

- `SRV:` defines hostname / port of the service offering node
- `A:` used to map the hostname to an IPv4 address
- `AAAA:` additionally used for IPv6 addresses
- `PTR:` used by devices to resolve other devices hostnames through their network addresses; used in mDNS to assign service instances to a service
- `TXT:` to propagate user-defined text, e.g., distribute presence in XMPP *Serverless Messaging* (see Section 4.3.1)

3.2 Cloud Services and Storage

Nowadays, cloud services enable users to access a reliable, personalized, highly scalable, secure, and fast infrastructure hosted by a third party. Several hosting providers like Amazon, IBM, and Microsoft operate large data centers, offering a wide range of cloud service products for moderate prices. Charges depend on the amount of storage or used CPU time and the amount of data transfer when using these services. Furthermore, a set of preconfigured images for cloud users is available so that a wide range of activities and tasks (e.g., web hosting, scientific analysis, online video processing) can be fulfilled. Cloud storage is a new concept based on cloud services to provide large amounts of online storage to support collaborative work without the need to install physical storage devices in the user’s datacenter or office. This reduces acquisition and maintenance costs while at the same time backup and reliability are offloaded to the hosting provider’s responsibility. Cloud storage is typically accessed through web interfaces [28].

[3] IANA List of Port Number Assignments [Online] `http://www.iana.org/assignments/port-numbers`

3.3 Discussion of Related Work

Combining sensors with cloud technology is a new trend as explained in [21, 25]. A main advantage is that time-critical sensor data can be processed instantaneously, but using a plain Wireless Sensor Network (WSN) for this has several drawbacks [9]: Energy is a big concern for typical sensor nodes, so data needs to be transported hop-by-hop to a network sink, causing delays in real-time data streams. The processing capabilities of resource constrained sensor nodes are also limited when compared with more powerful platforms. In addition, only small data packets are usually sent through a WSN, which might lead to a loss of detail or at least lags during the data analysis (cf. max. of 127 Bytes for IEEE 802.15.4 packets to min. 1280 Bytes for IPv6 packets). To this end, complex routing protocols are needed to send the aggregated data from one node hop-by-hop to the sink.

Still, WSNs and resource constrained devices are well suited for certain use cases (e.g., energy efficient structural health monitoring [31]). So instead of completely omitting WSNs, we favor a hybrid system composed of powerful yet portable hand-held devices with the ability to integrate autonomous sensors and networks of resource constrained devices. This integration is done by communicating with autonomous sensors over the portable hand-held devices (e.g., smartphones). They act as bridges, equipped with optional compatible wireless transceivers. This enables us to build a network with a backbone of powerful devices, where many smaller data streams can be collected and gathered, while we do not rely solely on low data rate protocols like IEEE 802.15.4 for the complete data exchange on-site. A comparable approach of a hybrid wireless communication system was described in [26], where the authors combined IEEE 802.11 and IEEE 802.15.4 to achieve monitoring as well as dispatch communication within a single hybrid system. We share this work’s basic idea, although our application scenario and the accompanied requirements differ.

A consequence of our design choice is a limitation of the system’s life time when compared with pure WSNs. The design choice reflects itself in our use case, since monitoring critical infrastructures and supporting post-disaster management is only a temporary challenge. Temporary in our case means days to weeks, instead of years or decades when compared with permanent structural monitoring [14]. Despite the shorter life time, we still gain the important benefit of flexibility and the ability to transfer and process larger data quantities in time-critical situations. With a backbone of portable and powerful hand-held devices, we avoid the limitation of sensing capabilities that is typical for energy and resource constrained sensor networks. More important than a long life time is to provide an uncomplicated system with an easy setup and comprehensible data access. Using only a WSN means that several decisions have to be made before deployment and activation of the system. Examples are decisions about which protocols and software to use, which sensor types to integrate, where to place the sensor nodes, how to access them in-field, or how to interconnect the sensor network to any external network (i.e., the Internet). In contrast, our design shall provide the flexibility to change these decisions alongside and after the deployment, thus separating it from standard WSNs.

A unique aspect of our approach is that we connect hand-held devices as well as autonomous sensors through a single protocol (i.e., XMPP). XMPP is for us the glue that holds different devices and networks together to enable a cross-border communication and collaboration to support post-disaster management stakeholders. In our design, hand-held devices and sensor devices are connected either directly to the Internet or through a one-hop communication via XMPP over another hand-held device to deliver a real-time data stream with a high resolution directly to the cloud service. Querying data from autonomous sensors at close range can be done through *XEP-0174*. In contrast to WSNs, where gateways [13] or complex middlewares [16] are needed for the interconnection of different protocols or networks, we favor XMPP as the sole communication protocol to offer clients a transparent access without the need for a middleware. This step could also solve the issue of non-standard software interfaces of sensors, already described in [25].

4 System Design

This section presents a flexible, reliable, cost-efficient, and widely applicable monitoring system for post-disaster management. The architectural design allocates the collaboration of XMPP-driven entities with cloud services to store sensor data and the interaction of observers with the cloud to retrieve the measured sensor data in infrastructure networks. Furthermore, an alternative access strategy for ad hoc networks is discussed. Essential for the architecture and the system design is a solution to access devices securely in hybrid networks through unique Jabber IDs (JIDs). This fundamental part is described below in detail, while evaluations and practical validations are outlined in Section 5.

4.1 Architecture of the System

The system consists of sensor-equipped portable devices, autonomous sensors with short range communication, stakeholders with different (sensor-equipped) devices and cloud services as depicted in Figure 2. Each device runs an XMPP software client through which it can access the XMPP network and publish sensed data. The cloud service manages a storage pool for the measured sensor data as well as several XMPP domains which facilitate an interconnection and data exchange between different organizations (e.g., rescue team, government, NGO). Data access for third parties / volunteers can be enabled by giving access to the sensor data storage.

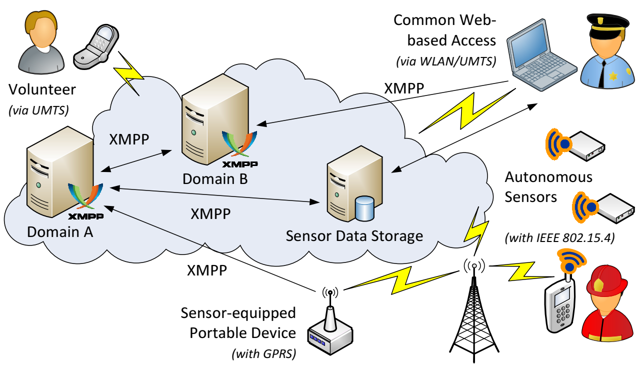

Fig. 2 System reference model

4.2 System Criteria

Based on the scenario outlined in Section 2, we expose various challenges applicable to post-disaster management. To enable a system classification we use the following criteria:

Standard protocols: XMPP and mDNS are established standards for detecting entities in ad hoc networks and for collaborative communication. In contrast to WSNs, where gateways are needed to link different protocols and networks, XMPP offers clients a transparent access to various devices, to different networks or to the sensor data storage. Moreover, XMPP implements the publish / subscribe paradigm, which is beneficial for scenarios where only changes in sensed data need to be transmitted to registered receivers. This helps to save bandwidth and energy [7] for the event distribution.

Temporary use: Emergency situations occur over a limited time period. Our proposed system is therefore designed to support run-time durations up to several weeks (refer to Section 3.3). A big advantage of short period monitoring is that commodity hardware can be used instead of dedicated wireless sensor network hardware with a long term (years) monitoring approach in mind.

Transportable entities: Sensor-equipped devices should be compact and transportable. We favor the use of commodity hardware running Linux instead of proprietary embedded hardware and accompanied operating systems to reduce maintenance and development costs. In the broader sense it offers us the possibility to run standards like mDNS [4] and XMPP on such devices, although existing protocol implementations might need to be adapted to include and integrate resource constrained sensor hardware.

Sensor groups: Sensor-equipped devices might consist of several detectors (e.g., compression, temperature, acceleration, humidity). Grouping of detectors will help handling a large number of these devices and will provide a better overview in an XMPP client software by using the XMPP roster management. This way, various entities become well-arranged and can easily be managed in a single sensor group.

Seamless integration: Integrating our system in a seamless way into the operating cycle of the crisis management and the participating rescue workers has a high priority because it provides additional information to ease decision making in critical situations. Sensor data access can therefore be realized by a standard XMPP chat client, ensuring a slim and fast implementation for various operating systems on many devices, ranging from desktop systems to hand-held devices.

Cost efficiency: Using embedded systems as sensor devices and commodity hardware like PCs, laptops, or smartphones to execute the monitoring and thus gain accumulated sensor data is an essential requirement for our system and will cut the costs significantly, because no special hardware must be developed. Autonomous sensors shall be integrated through standard interfaces (e.g., USB) and well-known communication systems (e.g., Bluetooth, IEEE 802.15.4). With the use of existing XMPP libraries, server and client development costs and test periods can be reduced, because designing, implementing, and intensively testing new underlying network protocols can be skipped.

Automatic configuration: User-driven configuration of the system should not be needed. Automatic configuration shall be an integral part of the system, so that the mobile entities of the system automatically detect neighbor entities in the ad hoc network and perform the initial registration to the XMPP server and the XMPP domain autonomously.

Network access: The system should support a wide range of network access technologies. A direct cloud access is preferred, using standard Internet protocols on top of Wireless Wide Area Networks (WWANs) like GPRS or UMTS. If an Internet connection can only be made by few devices, routing over Wireless Local Area Networks (WLANs) shall be used to share the Internet access through the XMPP network.

Scalability: The system’s performance is critical both for the sensor devices and the cloud services. It is important that the analysis system scales well with growing demands and increasing numbers of operations. Efficient and scalable implementations of cloud services are required.

Reliability: It is possible to avoid the loss of measured sensor data by storing it in the cloud-based data storage. The cloud service offers a statistical and more detailed view on the collected data. It can visualize the convergent sensed data from all devices and combine it into a comprehensive view, thus simplifying understanding of the received data. The whole event itself is logged in the data history and can be used for reviews or for overlaying functions in maps.

Data exchange: During emergency situations it might happen that several organizations from different kinds or countries have to work hand in hand. They also might need to access the same collected data. The exchange of data can be provided by granting access to both sensor devices as well as cloud services. Only a simple XMPP client is needed, which can be installed ad hoc for several platforms.

Expandability: A hardware independent integration of autonomous sensor devices and detectors into our system is realized through the use of XMPP. Our system can easily be extended with other devices that support XMPP. This will be beneficial for hardware solutions where only tiny or environment-specific detectors can be used and an embedded Linux does not fit into the hardware’s sparse memory.

4.3 XMPP Localization

XMPP entities can be located via a unique Jabber ID (JID). The uniqueness of the JID is guaranteed by the XMPP server when an entity is connected to its domain. In contrast to the infrastructure mode, the ad hoc XMPP network cannot ensure unique JIDs, because they are generated randomly in this case. This section introduces a solution where all entities in an XMPP network can be located via unique JIDs whether they are connected to an ad hoc or an infrastructure network. While entities with Internet access may connect directly to an XMPP server, users or devices in ad hoc networks without Internet sharing may find entities through the XMPP Extension Protocol (XEP) *XEP-0174 Serverless Messaging* [22]. An agent is used to connect ad hoc networks with the conventional XMPP infrastructure. The agent can have an integrated WWAN in addition to WLAN and thus act as a node with Internet access. In this connection a dedicated entity acting exclusively as a gateway is not necessary. With help from the agent each entity of the ad hoc network establishes a connection to the XMPP server. This has the advantage that each node authenticates itself against the server directly. Moreover, the compatibility with XMPP is guaranteed and any existing XMPP server infrastructure can be used. It also enables the use of any available JID (e.g., Google GTalk account [8]), thus eliminating the need for a system specific user identifier. Figure 3 illustrates the network structure of the localization system.

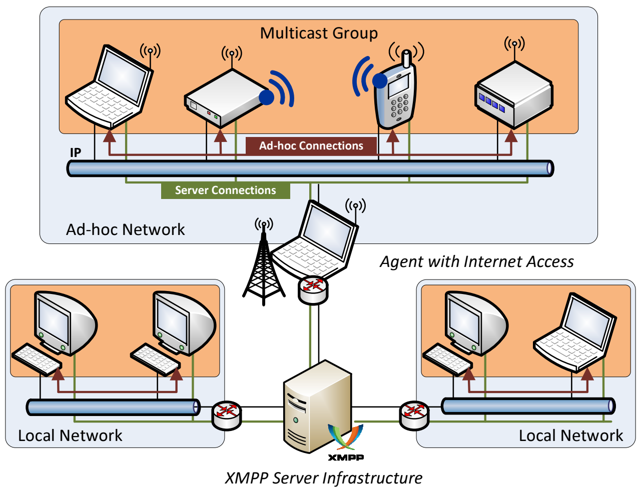

Fig. 3 XMPP network structure

4.3.1 XMPP Serverless Messaging

The mobile entities form an ad hoc network. Afterwards they join a multicast group on address `224.0.0.251` for IPv4 and address `ff02::fb` for IPv6. A user or a device advertises an online status with four different DNS records, namely a SRV, TXT, A, and PTR record. The value of the field name of the SRV record is set to the JID of the user or the according device. When a node joins the ad hoc network, it sends a PTR record (`_presence._tcp.local`) immediately to the multicast group. Now the other nodes reply with their information (one SRV, TXT, A, and PTR record per entity). If an entity wants to leave the network, it has to send a PTR record with TTL set to zero. Each node analyzes the released records inside the multicast group and each of them generates a list of entities based on the received records.

4.3.2 XMPP Ad Hoc Security

The localization service is provided by the XMPP base functionality and the following XEPs: *XEP-0166 Jingle*, *XEP-0174 Serverless Messaging*, and *XEP-0250 C2C Authentication using TLS*. Our system supports additional authentication and encryption for both ad hoc and client / server connections (enabled by default). The authentication of entities in ad hoc networks is realized by *XEP-0250* and the Secure Remote Password (SRP) [30] protocol. SRP was designed for client / server authentication. We modified the protocol to enable a pairwise authentication in ad hoc networks. For this, we use a shared password between both nodes during the authentication step. The exchange of transport addresses is realized over the authenticated and encrypted communication channel by *XEP-0166 Jingle*.

4.4 Access Strategies

The access to measured data can be achieved in two ways: (1) Access via standard chat clients that are nowadays preinstalled on most operating systems. (2) Common web based access over cloud services. Figure 2 depicts our system with several XMPP domains, sensor-equipped devices, the sensor data storage, and the described access strategies. Further enhancements for the data access could include easier access for volunteers and third parties. We think about providing access through common smartphone apps that volunteers can install optionally. This idea is still preliminary because security and accessibility questions are yet unsolved.

4.4.1 Data Collecting

Our portable sensor devices can be equipped with any compatible detector to enable data collecting, depending on the requirements of the current incident. It follows the principle of a construction or building block kit where all necessary components can be exchanged or upgraded. Default and preconfigured modules are GPS, an acceleration sensor, and a WWAN modem. This allows us to localize the sensor device over the Internet and to get the latest acceleration and sensor values on demand via the publish / subscribe paradigm of XMPP. *XEP-0174 Serverless Messaging* can be used as an alternative to read sensor data directly in the vicinity of a sensor device through an ad hoc network connection.

The publish / subscribe paradigm is a core functionality of our system to propagate new sensor data to the Internet. A threshold can be adjusted on every sensor-equipped device separately, depending on its integrated detectors. As soon as the threshold value is exceeded, sensed data is propagated to the Internet. This feature can also be disabled if a continuous data stream is requested. We try to reuse the most common XEPs for the message format, to comply with existing XMPP clients because some clients do not support all existing XEPs. The *XEP-0163 Personal Event Protocol* allows for a user-defined event propagation to easily deploy personal event services across the XMPP network. Thus additional information can be sent from a sensor device to the Internet and can then be viewed by an XMPP client supporting *XEP-0163 (Personal Event Protocol)*.

4.4.2 Data Access

Every sensor device that is connected to the Internet will eventually also propagate its sensed data. Access to this data is allowed through XMPP, which is covered by today’s usual security standards. Equipped with different detectors, sensor devices can be accessed via different criteria. On one side, sensor devices will be grouped by their current GPS position. This way it is possible to access only sensor devices in a predefined radius around a selected target. On the other side, different measured data will be grouped as well, so that a powerful and hierarchical filtering of sensor devices can be done. The filtering request will also support known mathematical operations like *AND*, *OR*, and *NOT*.

4.4.3 Communication Flow

Figure 4 depicts an exemplary communication flow:

I: Devices start connecting to the Internet (I.1) via non-disrupted infrastructure or via portable access points (in case of destroyed infrastructure). Afterwards they register themselves in an XMPP domain (I.2).
II: Sensor-equipped devices join a virtual group, depending on their GPS position and attached detectors.
III: Devices start collecting data and send it to the Internet, where it is stored in the cloud-based sensor data storage.
IV: Fall-back and direct access via *XEP-0174*: devices can be accessed directly by rescue workers in the outer quarter in case central infrastructure fails and to integrate autonomous devices over Bluetooth and IEEE 802.15.4.

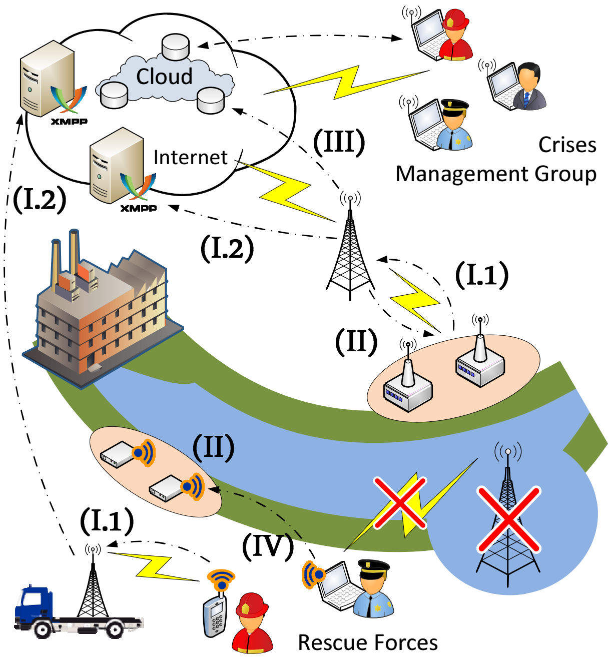

Fig. 4 Exemplary system communication flow

The fall-back mechanism of the system is based on *XEP-0174 Serverless Messaging* [22]. It allows for a direct access by the rescue workers in the outer quarter if: the Internet connection breaks down (natural hazard), a dedicated Internet connection for the current situation is not recommended, for areas where radio resources are short-staffed, or for accessing autonomous sensors over short-range communication technologies (e.g., Bluetooth, IEEE 802.15.4). This connection over rescue workers also enables a direct forwarding of data from autonomous sensors into the cloud.

The configuration of the portable sensor-equipped devices can be manifold and freely selectable. Our basic setup will offer two variations: modules with a dedicated (a) and with a shared (b) Internet connection. Refer to Figure 5 for a detailed illustration. A dedicated device module will have its own Internet connection through WWAN technologies like GPRS, UMTS, or LTE. This module facilitates an independent and uncomplicated installation of each dedicated sensor-equipped device. XMPP enables a transparent communication with each dedicated module addressed by its GPS position, measured data, and configuration, without requiring complex server infrastructure in the outskirts of an emergency incident. Beside this point each dedicated module can be placed side by side to monitor a detailed area or be spread out to cover a critical infrastructure in whole.

A shared or joint-use module builds up a group of many sensor-equipped devices to monitor a specific part of the disaster site. All sensor-equipped devices in this group will share one Internet connection established by one configured agent. The agent facilitates Internet access through its integrated WWAN device for all other devices in the group and it is responsible for the forwarding of all measured sensor data of the group into the Internet. The advantage of this strategy is that costs can be cut down by integrating only one WWAN module and by using only one WWAN data channel for all group devices while the access to the measured sensor data is still transparent with XMPP.

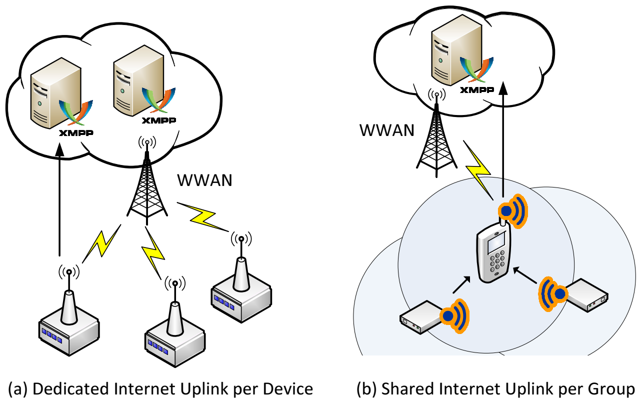
XMPP
XMPP
WWAN
(a) Dedicated Internet Uplink per Device
XMPP
WWAN
(b) Shared Internet Uplink per Group

Fig. 5 Dedicated (a) and Shared (b) sensor-equipped device modules

4.4.4 Data Storage and Aggregation (Cloud)

As mentioned earlier, the sensor data will also be stored in the sensor data storage for further analysis through cloud services (data warehouse principle). The cloud consists of the sensor data storage and an analysis tool for the collected sensor data, optionally a dedicated XMPP server. Our system normally uses public XMPP servers[4] because they are well maintained, available in manifold configurations, and free to use. Under certain circumstances it can make sense to deploy a private XMPP server, for which a pre-configured XMPP component is ready to be used. The sensor data storage acts as a normal XMPP client inside the network that logs all measured sensor data into a database. With the calculation power of the cloud it is possible to analyze the coherence of previous events and create a forecast on the impacts of future events. The cloud services and the storage pool make it possible to interact with the measured data in many ways, e.g., display the sensor results on a map or calculate statistical results. Scalability is the key feature of the cloud system so there is no doubt about restricted resources. Costs can be reduced due to the fact that the cloud service is only paid per use. The whole analysis component can be frozen if not needed and started again on demand.

With the collaboration of sensor-equipped devices and cloud services, a new set of properties can be achieved:

– The number of used sensor device modules and the number of users with access to these devices scale freely;
– A chronological progression of past sensor data can be viewed and inspected from each control point along with the current measurements;
– Remarkable data aggregations like frequency distribution or mean can be calculated in the cloud and results can be provided to other users;
– If the Internet connection breaks down (natural hazard) the sensor devices can be accessed directly by rescue forces in the outer quarter via *XEP-0174*;
– Sensor device modules from different organizations can be interconnected with XMPP, while providing a continuous and transparent view on the measured data.

Results generated by cloud services can be viewed through a common web based interface. Users will have the choice between a simple and clean XMPP client, to view currently measured data directly, or a feature rich web view applied by the cloud service with prepared sensor data.

4.4.5 Collaboration Support

Creating a joint task force between several rescue teams from different kinds or countries during an emergency situation can be provided by granting access to the sensor devices or to the cloud service. This will enable a direct exchange and analysis of collected data between the crisis management and the rescue forces. To support collaboration, new tasks for each rescue team can be adjusted ad hoc and managed by the crisis management team as a joint task. Communication groups and chat rooms (realized with *XEP-0045 Multi-User Chat*) offer the possibility to coordinate joint forces easily, classified and task specific. Example situations could be exclusive groups for fire fighters, police men, or ambulance teams to interact undisturbed in critical situations while coordination information could be announced via a chat room to all rescue forces. Therefore, only a simple XMPP client is needed to enable a range of collaboration options, hence reducing software costs.

While XMPP chat rooms enable a rapid interconnection by using commodity hardware like smartphones or laptops, involving and supervising volunteer citizens in emergency situations can now be simplified. The volunteers only need to use an ordinary XMPP software client to support rescue forces, e.g.: by alerting them about dangerous spots, by informing them about the geo-location of affected humans, to receive tasks from the crisis management, to forward collected data from sensor devices in the outer quarter, or to interact with other volunteers. If the necessary software is not pre-installed on the smartphone or tablet of a volunteer, an XMPP client can simply be downloaded and installed from the Internet or from a public app store. Furthermore, a communication channel with information on the progress of the current rescue actions can be established through a public XMPP chat room. The crisis management can publish aggregated data as formatted news directly from the cloud services to the communication channel while affected humans could report their current geo-location and health state through this channel back to the crisis management. Again, only a simple XMPP client is needed here.

4.5 Sensor Device Prototypes

The sensor devices can be equipped with several detectors (e.g., compression, temperature, radiation, acceleration), depending on the actual incident. The following subsections introduce two prototypes for the realization of our XMPP-based post-disaster monitoring system. The first prototype depicts a smartphone with included or externally attached sensors, as shown in the reference model from Figure 2. It also resembles the dedicated sensor-equipped device from Figure 5 (a). The second prototype represents the group of autonomous resource constrained sensor devices, named autonomous sensors in Figure 2. This prototype is connected through IEEE 802.15.4 to our first prototype. These two prototypes demonstrate that XMPP can be used successfully for post-disaster management scenarios with diverse devices.

[4] Public XMPP servers [Online] http://www.jabberes.org/servers/servers_by_times_online.html

4.5.1 Sensor-equipped Smartphone Prototype

In our application scenario, the smartphone can be used to detect dynamic loads on buildings while sending the measured data from the integrated accelerator to the Internet. The device could simply be attached to the walls of a building to gain measurements. Our prototype is based on an HTC HD2 smartphone [11]. The smartphone runs Windows Mobile 6.5, supports GSM and UMTS, has an integrated GPS receiver, and includes a proximity and a G-sensor. We chose the HTC HD2 as a prototype device because it is handy, can run up to max. 15 days, assists several sensors, and supports running `C/C++`-written applications on its operating system. The smartphone can easily be extended with additional devices through several standard interfaces like microUSB or the SDIO connector. Detectors that are externally connected to the smartphone will be coupled via our specific *XEP-0174*-based communication stack for resource constrained devices (refer to Subsection 4.5.2 and Figure 6). The HD2 smartphone is a good example for running our XMPP localization solution and submitting the currently measured sensor data to the Internet. Furthermore, it meets our system criteria introduced in Subsection 4.2.

We resort to standardized technologies like XMPP, TLS, SRP, DNS-SD, and mDNS for our work, available for various programming languages, platforms, and systems. Our criteria for choosing libraries were: support for a wide range of operating systems (Windows Desktop / Mobile, Linux) and platforms (x86, ARM); written in `C/C++`; and available under a free software license. We assume that there is at least one `C/C++`-compiler available for each platform. We chose the libraries *mDNSResponder* [1], *gloox* [24], and *OpenSSL* [19] based on the mentioned criteria. mDNS and DNS-SD are provided by mDNSResponder while gloox provides a client site implementation of the XMPP Core and OpenSSL implements the TLS / SSL protocol.

4.5.2 Autonomous Sensor Prototype

The autonomous sensor prototype is meant to be an extension for the sensor-equipped smartphone. We want to promote using such autonomous sensors for scenarios where the area cannot be safely reached or where rescue forces are not allowed to stay over a longer time period (e.g., next to radioactive zones). Autonomous sensors also allow us to attach an even wider array of detectors and sensor modules, since they provide a multitude of connection ports (e.g., I^2C, ADCs, TwoWire bus). We chose the Zolertia Z1[5] as our prototype, which is based on a MSP430 MCU for ultra-low power consumption. It represents a typical resource constrained device with limited memory (92 kB of ROM / 8 kB of RAM), it runs the embedded operating system Contiki [6], it provides built-in sensors (temperature, accelerometer, battery level), and it offers a variety of connectors. The Zolertia Z1 offers communication via an IEEE 802.15.4 compliant RF transceiver (CC2420 radio chip) and via an microUSB connector. We can hence extend the HD2 smartphone with a microSD card with an IEEE 802.15.4 compliant transceiver or connect the Zolertia Z1 directly with a microUSB cable (USB host drivers are needed) to establish a wired or wireless connection between our two prototypes. In contrast to Bluetooth connections there is no complicated pairing mechanism necessary, because nearby devices simply appear in the networking environment through announcements by mDNS and DNS-SD. Another advantage of the Zolertia Z1 is its expandability and exchange-ability via external detectors, e.g.: IR distance, gas pressure, AMP current, magnetic, precision light, or sonar sensors[6].

Resource constrained embedded devices typically provide limited memory sizes and slow microcontrollers. The used operating system as well as the used XMPP, mDNS, and DNS-SD implementations hence need to be memory-efficient and extremely lightweight. Instead of using an embedded Linux, we favor the lightweight and highly portable open source operating system Contiki [6] for embedded and memory-constrained hardware. Contiki runs on small networked sensor boards with a typical configuration of 40 kB of ROM / 2 kB of RAM and it provides an embedded IPv4 / IPv6-compatible stack, which allows TCP/IP connections to systems including UDP and TCP support.

Contiki supports the programming of applications in `C` while offering a `C`-compiler for a variety of embedded processors. Libraries developed for desktop or mobile system cannot be used in this case because these are mostly written in `C++` and such libraries are too heavy for resource constrained hardware. We developed our own tiny *XEP-0174*-based communication stack consisting of a mDNS / DNS-SD service and a *XEP-0174* client for Contiki as depicted in Figure 6. It operates together with Contiki's integrated low-power uIP stack and fits into the memory of the Zolertia Z1.

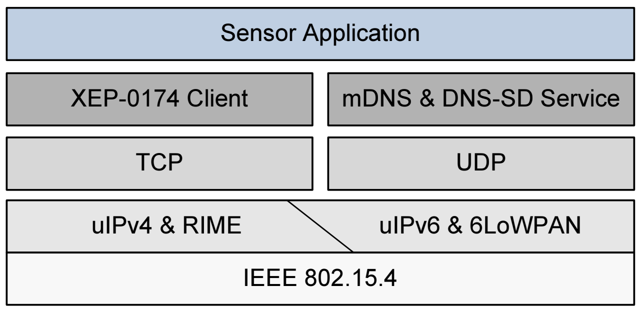

Fig. 6 *XEP-0174*-based communication stack with test application

[5] Zolertia Webpage [Online] http://www.zolertia.com/

[6] List of Z1 Sensors [Online] http://zolertia.sourceforge.net/wiki/index.php/Z1_Sensors

The μXMPP project [10] was used as a starting point for the development of the *XEP-0174* client because it offers a parser and simple XMPP message exchange. We added a TCP handler to manage incoming TCP connection requests and added support to accept an opening XML stream from other entities in the network. The mDNS and DNS-SD services are realized in a memory-efficient manner through in-place processing: the parsing and generating of DNS messages is directly done in the IP buffer of Contiki to reduce the necessary buffer size. Our implementation features the resolving of hostnames and services as well as the registering, removing, and updating of services in the network [15]. At the moment, no secure connection between an external and autonomous sensor and a smartphone is provided. We plan to use MatrixSSL[7], an embedded SSL and TLS implementation designed for small footprint applications and devices, for future extension of our implementation.

5 Performance Analysis and Practical Validation

This section presents our initial evaluations and performed validations. The smartphone prototype and two connected test cases are analyzed in Subsection 5.1. A verification and initial performance evaluation for the autonomous sensor prototype are performed in Subsection 5.2. Each part closes with a discussion of results.

5.1 Sensor-equipped Smartphone Prototype Performance

This section covers the performance analysis of the system's fall-back mechanism based on *XEP-0174* and the scalability of mDNS-driven ad hoc networks. We analyze the performance based on testing a node's network switching time from infrastructure to ad hoc mode and vice versa. We can hence determine how fast our devices switch between both modes, which is important for our hybrid network scenario. Tests were performed for both directions (a node joins and leaves the ad hoc network). In addition the data traffic that is generated by DNS-SD is estimated. This estimation enables an evaluation of our system's scaling performance for the maximum number of participants in the ad hoc network because it depends on the amount of sent DNS-SD messages.

Our test environment consisted of a smartphone with an integrated accelerometer (`node A`) and a standard laptop (`node B`) to reproduce the application scenario for the device prototype from Section 4.5.1. This reduced test environment allows us to determine the minimal switching time and to evaluate the maximum scaling factor of mDNS while the competing network access of other devices can be neglected. The development of the test software for the smartphone prototype was very short and cost effective, because existing libraries could be reused for all involved devices as described in Section 4.5.1. We chose the XMPP server from the *xmppnet project*[8] for our test, which is available under the domain `xmppnet.de` on the standard XMPP port 5222. For every node an XMPP account with a unique JID was created. Both nodes formed an IEEE 802.11-based ad hoc network. No encryption and no authentication were used to measure only the real network switching time. The distance between the nodes was `five` meters. During the tests a line of sight between the two nodes was ensured. Node A acted like a node that joined and left the network. In test case 1 node A established a UMTS connection after leaving the ad hoc network. Over this established Internet connection the node connected to the public XMPP server. In the reciprocal test case 2, node A joined the ad hoc network and disconnected from the server before it disconnected the UMTS connection. Node B remained fixed in the WLAN ad hoc network and offered node A a permanent chance to join the ad hoc network. The network switching time was measured by monitoring the change of presence from node A in the ad hoc network and on the server. For this purpose node B was permanently connected to the Internet through a high speed uplink. The network switching time is composed as follows:

1. Time of setting up the UMTS connection;
2. Time to register at the XMPP server;
3. Time to transmit the presence status to the XMPP server;
4. Delays that occur by handling further XMPP messages to the XMPP server and vice versa.

5.1.1 Test Case 1: Node left the Ad Hoc Network

In test case 1 node A left the ad hoc network and created a UMTS connection. Afterwards the node authenticated itself against the XMPP server. Node B measured the time between leaving the ad hoc network and the change of presence to the state `offline` of node A on the server. Before node A connected to the server it left the ad hoc network.

Figure 7 shows the gained measurements. Each bar, depicting one of the ten performed test runs, is subdivided in the composition of the network switching time as described above. The network switching takes 11.7 *s* in average. It is mostly influenced by the delay of establishing a connection to the server. It causes the biggest part of the network switching time as shown in Figure 7. The reason is the unsteady mobile network connection provided by the UMTS network. Furthermore, a strong variation of the network switching time was noticed. The connection establishment to the server varies in a range of 1 *s*. We assume that the UMTS network is another cause for the strong variation. The given value for connection establishment also contains the delay for the request to the DNS server which maps the

[7] MatrixSSL [Online] http://www.matrixssl.org/

[8] XMPPNet Project [Online] http://www.i-pobox.net/

hostname onto an IP address. DNS request processing takes between 100 *ms* and 300 *ms*.

Surprisingly high are the further delays in each test run because they take around 13.2 % from the overall network switching time. These delays are established at the server by the processing of presence information as well as in the test application itself. Additionally these delays consist of inaccurate estimations of other delays, e.g., the Round Trip Time (RTT) on node B’s side may be considered too low for the particular test run. Other inaccuracies in the test are based in the test application itself. An example is the delayed notification of the currently activated UMTS interfaces. This delayed notification is one reason why an earlier initiation of a server connection might not be possible.

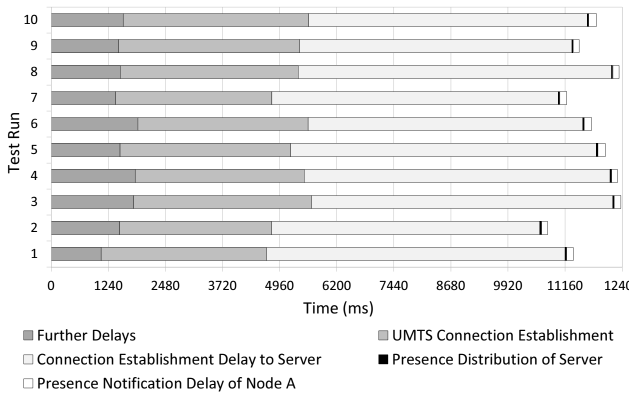

Fig. 7 Results of *Test Case 1* - Node left the ad hoc network

5.1.2 Test Case 2: Node joined the Ad Hoc Network

In the second test case node A joined the ad hoc network. Node B started the timer as soon as the server set the presence status to the offline state for node A. Once node B received the initial PTR record of node A the timer was stopped. A disconnect from the server was performed before joining into the ad hoc network was started.

The results of the measurements from test case 2 are depicted in Figure 8. The results depend on the length of the WLAN connection establishment delay and the processing delays of the test application itself. The activation of the WLAN interface and the joining into the WLAN-driven ad hoc network take the largest part of the network switching time with approx. 3 *s*. This part includes the time for the allocation of an IP address using dynamic configuration of IPv4 Link-Local Addresses. During this part it has to be verified that the self-chosen IP address is not being used by another node in the same ad hoc network. This verification step lasts between 1 and 2 *s* (refer to [5]). The additional delays occur in the test application itself, e.g., the delayed notification of currently activated WLAN interfaces, so that an earlier joining of the multicast group might not be possible.

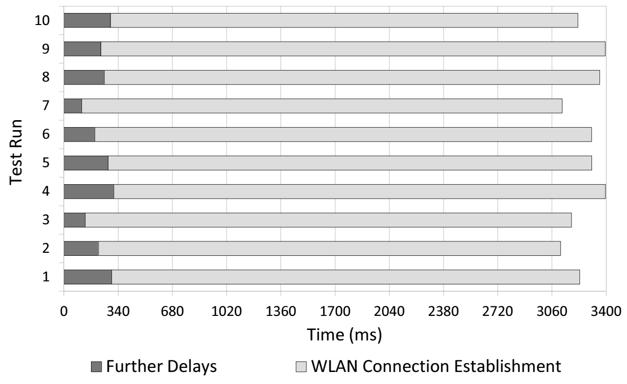

Fig. 8 Results of *Test Case 2* - Node joined the ad hoc network

5.1.3 Estimation of Data Traffic in mDNS

Participants signalize the join / leave into / from the multicast group by using DNS-SD messages. The generated data traffic is estimated to evaluate how the system scales with a growing amount of participants. The involved protocol layers and their parts of a message are clearly represented in Table 1. The protocol layers (DNS, UDP, IP) below DNS-SD use 40 Bytes constantly for every message.

Table 1 Fraction of sent DNS-SD messages (in Byte)

	PTR query	**DNS response**	**PTR leave**	**TXT update**
DNS-SD	26	180 - 500	78 - 141	150 - 250
DNS	12	12	12	12
UDP	8	8	8	8
IP	20	20	20	20
Total	66	220 - 540	118 - 181	190 - 290

We determine that the joining of a node in a multicast group with *n*-active participants causes data traffic between $(66+220+n*220)$ Bytes and $(66+540+n*540)$ Bytes. If, for example, a node joins a group with 50 participants, a maximum of 27.6 kBytes needs to be transferred in the worst case. This amount of traffic is acceptable according to our evaluations. We further used compression methods to reduce the data traffic. Two methods are available: the *Known-Answer Suppression* [4, Sec. 7.1] and the *Duplicate Question Suppression* [4, Sec. 7.3] method. Both are described in the mDNS specification and illustrated shortly below.

The known-answer suppression method enables a reduction of the amount of answers. Usually every node sends an answer to a given request (e.g., a node asks for other nodes with XMPP support). By using this method, a node sends a response for a group of many nodes (including himself), thus reducing the number of necessary responses to gather information about the whole network. To enable this method, each node caches the service offerings that are published in the ad hoc network. If a node has cached the answer to a request then the node adds this answer to its own. If the other nodes recognize this they will not send their own answers. This method assumes that each node waits a randomly chosen time before it answers a request. For a group of 50 participants, 49 answers may be reduced in the optimal case.

The duplicate question suppression method reduces the amount of requests. When a node sees a request that matches its own, it will assume this request as its own. In that way less PTR query messages are send, because sending redundant DNS response messages is prevented. Again, each node has to wait a randomly chosen period before it can send its request. Practical evaluations of the impact of these two methods on data traffic in ad hoc networks are future work.

5.1.4 Discussion of the Performance Analysis Results

In conclusion we can say that the XMPP implementation can be used to locate users by their unique JIDs in infrastructure and ad hoc networks. Comparing the measured switching times from joining and leaving the ad hoc network, we can see that the largest amount of time is consumed by entering the UMTS network and the successive server connection establishment delay. These situations occur only when the system is powered-on and they do not affect the work flow of our entire system after the initialization. The XMPP component works as expected with a low delay rate by transferring the presence status of every node. In addition, an estimation of the scaling performance of mDNS for the maximum number of participants in the ad hoc network was given.

5.2 Autonomous Sensor Prototype Verification

The autonomous sensor prototype has been implemented in a proof-of-concept application under Contiki to work with our *XEP-0174*-based communication stack and to analyze its memory footprint. Energy efficiency was not scope of the test, since the low-power quality of uIP was already proven in [12, Sec. 10]. The sensor node test application addresses the built-in detectors and combines them with our communication stack as shown in Figure 6. If a threshold value of a detector is exceeded, an appropriate *XEP-0174* presence status will be send from the sensor node to the network via our communication stack. The interaction with our test application running on the Zolertia Z1 is done via *XEP-0174* messages from an ordinary XMPP client. Threshold values for the detectors can be adjusted through the XMPP client, while current values of each available detector can be received and the battery state can be monitored as well. The concept is inspired by *XEP-0050 Ad-Hoc Commands*. This XEP allows users to initiate a command session and to interact with an automated process through the XMPP client. Our test application announces a command list when an XML stream was opened with the user’s XMPP client. A command can be executed by the user by sending a XMPP message containing the command and its parameters.

Our experiment runs with Contiki 2.5 on the Zolertia Z1 platform, which we connect to a computer running Linux via the Serial Line Internet Protocol (SLIP). For testing we connected the device either directly (simulating a wired connection via USB) or indirectly while using a 1-hop network with static routes (simulating a wireless connection via 802.15.4 over a wired Z1 via USB) to the computer. The computer ran an unmodified Pidgin[9], a universal and multi-platform XMPP chat client, and Avahi [27], an implementation of the mDNS and DNS-SD specifications for Unix operating systems. Avahi was pre-configured with *enable-reflector=yes* and *allow-point-to-point=yes* to rebroadcast the mDNS and DNS-SD messages from SLIP to all available Ethernet interfaces of the computer. Pidgin recognizes the incoming DNS-SD-based presence messages and sorts the corresponding contact to the Bonjour group of the user’s buddylist. If the sensor node appears in the buddylist, the user can start a communication with the external sensor node via standard-compliant *XEP-0174* messages. The Z1 itself sends a message to Pidgin at the beginning of the conversation containing a list of available commands. With these commands the user is able to interact directly with the Z1 to get the current or last cached sensor readings through the XMPP client.

5.2.1 Memory Footprint

Our test focuses on the memory footprint of our test application because a memory-efficient implementation is essential to perform properly on resource constrained devices. The code is compiled with msp430-gcc (GCC) 4.4.5 for the Zolertia Z1. We used the default settings of the Zolertia Z1 configuration, no optimizations were made for Contiki and its integrated IP stack. The mDNS/DNS-SD service and the *XEP-0174* client can be used with IPv4 and IPv6.

Table 2 shows the detailed memory footprint of all components of the test application. The mDNS / DNS-SD component uses 3.82 kBytes of ROM / 0.3 kBytes of RAM for IPv4. Minimal larger buffers are needed for sending and storing registered services with IPv6 support, which will cost additional 0.07 kBytes of ROM for the mDNS/DNS-SD service. The total memory usage for the sensor node test application is 41.64 kBytes ROM and 5.95 kBytes RAM while the *XEP-0174* client (prepared for at least one TCP connection) takes 6.69 kBytes of ROM / 0.43 kBytes of RAM and the sensor test application needs 3.81 kBytes of ROM / 0.11 kBytes of RAM. It should be noted that there is only a small

[9] Pidgin [Online] http://pidgin.im/about/

difference for the memory consumption of these two applications with IPv4 or IPv6 because the code only differs in the used IP-address length while the rest is handled for each IP version by Contiki’s IP stack.

Table 2 Memory footprint of our *XEP-0174*-based communication stack operating under Contiki for the Zolertia Z1 (in kByte)

Component	**ROM**	**RAM**
Sensor test application	3.81	0.11
XEP-0174 client	2.87	0.13
mDNS & DNS-SD	3.82	0.3
IPv4 stack (UDP & TCP)	10.77	1.97
Contiki	20.37	3.44
Total	41.64	5.95

5.2.2 Discussion of the Verification and Analysis

The memory footprint of our overall *XEP-0174*-based communication stack (including the mDNS / DNS service, the *XEP-0174* client, the IPv4 stack, and the Contiki OS) is less than 37.9 kBytes ROM and 5.9 kBytes RAM. The current implementation of the *XEP-0174* client is yet not optimized and should offer potential to minimize the memory consumption in further stages. As such, a mDNS / DNS-SD service and a *XEP-0174* client were verified to run on resource constrained devices like the Zolertia Z1.

One reason for the slim implementation is that we do not use an XML parser for the XMPP message parsing. Instead, we simply used a string compare method for incoming messages in the *XEP-0174* client. This parsing approach turned out to be not flexible enough during our tests because we noticed that XMPP servers and clients differ in the use of separators and upper- and lowercase spelling in XML which breaks a simple string compare. In future work, we want to implement a lightweight XML parser for XMPP messages in Contiki because string comparison is too failure-prone when compared with regular parsers.

6 Conclusions

This work presented an XMPP-driven system that combines sensor-equipped devices with cloud services to support post-disaster management. We showed that different classes of devices and networks can be connected by using a standardized collaborative communication protocol (XMPP) without the need for a middleware. We introduced our system design and the characteristics that it should fulfill and evaluated them under consideration of various criteria, which helped us to distinguish our work from existing research in the area of wireless sensor networks. Through the implementation and evaluation of two prototypes, we gathered new insights on the applicability of XMPP for the JID-based localization in ad hoc and infrastructure networks. As a result of the implementation process, we acquired a secure, bi-unique, and transparent access to all involved communication partners in our system and we validated that XMPP can be used for resource constrained devices running Contiki. We also described the integration of the cloud system as a data storage and data aggregation center, which is scalable and ensures the reliability of stored sensor data. With the help of the cloud system and our introduced access strategies rescue forces, crisis management, and volunteers can work hand in hand while information of the emergency situation can be provided to affected humans at any time.

Further work resides in extensive practical testing and a detailed analysis of the impact of autonomous sensor devices on our system architecture. Tiny sensors can be used in large quantities, which might make them highly influential on the overall communication flow of our system. To perform practical studies of such behavior, we depend on network simulation frameworks, simply because we alone are not capable of building larger setups. An open issue for such simulation studies is the realistic behavior of the autonomous sensor nodes. We prefer to run exactly the same code from our real-life implementations inside the simulation framework to omit the typical problems of abstract simulation models and hence inaccurate results. To achieve this, we plan to integrate the used prototypes together with complete Contiki instances running our *XEP-0174*-based communication stack into a simulation environment, enabling a variation of co-simulation in the end. This will eventually enable us to directly simulate the system behavior in close relation to the expected real-life performance of our implementation for resource constrained devices.

References

1. Apple Inc (2011) mDNSResponder. http://www.opensource.apple.com/tarballs/mDNSResponder/, Accessed 21-02-2012
2. Bernstein D, Vij D (2010) Intercloud Directory and Exchange Protocol Detail Using XMPP and RDF. In: Proceedings of the 6th World Congress on Services (SERVICES 2010), IEEE Computer Society, pp 431–438, DOI 10.1109/SERVICES.2010.131
3. Cheshire S, Krochmal M (2011) DNS-based Service Discovery. Internet Draft, IETF
4. Cheshire S, Krochmal M (2011) Multicast DNS. Internet Draft, IETF
5. Cheshire S, Aboba B, Guttman E (2005) Dynamic Configuration of IPv4 Link-Local Addresses. Request for Comments 3927, IETF
6. Dunkels A, Gronvall B, Voigt T (2004) Contiki - A Lightweight and Flexible Operating System for Tiny

Networked Sensors. In: Proceedings of the 29th IEEE Conference on Local Computer Networks (LCN 2004), IEEE, pp 455–462, DOI 10.1109/LCN.2004.38

7. Eugster PT, Felber PA, Guerraoui R, Kermarrec AM (2003) The Many Faces of Publish/Subscribe. ACM Computing Surveys Journal (CSUR) 35(2):114–131, DOI 10.1145/857076.857078
8. Google Inc (2012) Google Talk. http://www.google.com/talk/, Accessed 21-02-2012
9. Heidemann J, Silva F, Intanagonwiwat C, Govindan R, Estrin D, Ganesan D (2001) Building Efficient Wireless Sensor Networks with Low-Level Naming. In: Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP 2001), ACM, pp 146–159, DOI 10.1145/502034.502049
10. Hornsby A, Bail E (2009) uXMPP: Lightweight Implementation for Low Power Operating System Contiki. In: Proceedings of the Conference on Ultra Modern Telecommunications Workshops (ICUMT 2009), DOI 10.1109/ICUMT.2009.5345594
11. HTC America Inc (2011) HTC HD2 Specifications. http://www.htc.com/us/products/t-mobile-hd2#tech-specs, Accessed 21-02-2012
12. Hui JW, Culler DE (2008) Ip is dead, long live ip for wireless sensor networks. In: Proceedings of the 6th ACM conference on Embedded network sensor systems, ACM, New York, NY, USA, SenSys ’08, pp 15–28, DOI 10.1145/1460412.1460415
13. Isomura M, Decker C, Beigl M (2005) Generic Communication Structure to Integrate Widely Distributed Wireless Sensor Nodes by P2P Technology. In: Adjunct Poster Proceedings of the 7th Conference on Ubiquitous Computing (Ubicomp 2005)
14. Kim S, Pakzad S, Culler DE, Demmel J, Fenves G, Glaser S, Turon M (2006) Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks. Tech. Rep. UCB/EECS-2006-121, EECS Department, University of California, Berkeley
15. Klauck R, Kirsche M (2012) Bonjour Contiki: A Case Study of a DNS-based Discovery Service for the Internet of Things. In: Li X, Papavassiliou S, Ruehrup S (eds) ADHOC-NOW 2012, Lecture Notes in Computer Science, vol 7363, Springer Berlin, pp 317–330
16. Kusznir J, Cook DJ (2010) Designing Lightweight Software Architectures for Smart Environments. In: Proceedings of the 6th Conference on Intelligent Environments (IE 2010), IEEE Computer Society, pp 220–224, DOI 10.1109/IE.2010.47
17. Mockapetris P (1987) Domain Names - Implementation and Specification. Request for Comments 1035, IETF
18. Morreale P, Qi F, Croft P (2011) A Green Wireless Sensor Network for Environmental Monitoring and Risk Identification. International Journal of Sensor Networks 2011 10(1/2):73–82
19. OpenSSL Development Team (2012) OpenSSL Project. http://www.openssl.org/, Accessed 21-02-2012
20. Peng J, Zhang X, Lei Z, Zhang B, Zhang W, Li Q (2009) Comparison of Several Cloud Computing Platforms. In: Proceedings of the 2009 2nd Symposium on Information Science and Engineering (ISISE 2009), IEEE Computer Society, pp 23–27, DOI 10.1109/ISISE.2009.94
21. Rajesh V, Gnanasekar JM, Ponmagal RS, Anbalagan P (2010) Integration of Wireless Sensor Network with Cloud. In: Proceedings of the 2010 Conference on Recent Trends in Information, Telecommunication and Computing (ITC 2010), IEEE Computer Society, pp 321–323, DOI 10.1109/ITC.2010.88
22. Saint-Andre P (2008) XEP-0174: Serverless Messaging. Standards track, XMPP Standards Foundation
23. Saint-Andre P (2011) Extensible Messaging and Presence Protocol (XMPP): Core. Request for Comments 6120, IETF
24. Schroeter J (2011) Gloox. http://camaya.net/gloox/, Accessed 21-02-2012
25. Shu L, Hauswirth M, Cheng L, Ma J, Reynolds V, Zhang L (2008) Sharing Worldwide Sensor Network. In: Proceedings of the 2008 Symposium on Applications and the Internet (SAINT 2008), IEEE Computer Society, pp 189–192, DOI 10.1109/SAINT.2008.22
26. Tao P, Xiaoyang L (2011) Hybrid Wireless Communication System Using ZigBee and WiFi Technology in the Coalmine Tunnels. In: Proceedings of the 2rd Conference on Measuring Technology and Mechatronics Automation (ICMTMA 2011), IEEE Computer Society, pp 340–343, DOI 10.1109/ICMTMA.2011.372
27. The Avahi Team (2011) More About Avahi - Details about mDNS, DS-DNS and Zeroconf. http://avahi.org/wiki/AboutAvahi, Accessed 21-02-2012
28. Wang Y, Wen X, Sun Y, Zhao Z, Yang T (2011) The Content Delivery Network System Based on Cloud Storage. In: Proceedings of the 2011 Conference on Network Computing and Information Security (NCIS 2011), IEEE, pp 98–102, DOI 10.1109/NCIS.2011.28
29. Weis G, Lewis A (2009) Using XMPP for Ad-hoc Grid Computing - An Application Example using Parallel Ant Colony Optimization. In: Proceedings of the Symposium on Parallel & Distributed Processing (IPDPS 2009), IEEE, DOI 10.1109/IPDPS.2009.5161115
30. Wu T (2000) The SRP Authentication and Key Exchange System. Request for Comment 2945, IETF
31. Xu N, Rangwala S, Chintalapudi KK, Ganesan D, Broad A, Govindan R, Estrin D (2004) A Wireless Sensor Network for Structural Monitoring. In: Proceedings of the 2nd Conference on Embedded Networked Sensor Systems (SenSys 2004), ACM, pp 13–24, DOI 10.1145/1031495.1031498