Mobile XMPP and Cloud Service Collaboration: An Alliance for Flexible Disaster Management

Ronny Klauck and Jan Gaebler
Innovations for High Performance Microelectronics (IHP)
Leibniz-Institute for Innovative Microelectronics, Germany
Email: {klauck, gaebler}@ihp-microelectronics.com

Michael Kirsche and Sebastian Schoepke
Computer Networks and Communication Systems Group
Brandenburg University of Technology Cottbus, Germany
Email: {mkirsche, seb}@informatik.tu-cottbus.de

***Abstract*—Recent crises like the Fukushima incident in Japan show that there is a demand for flexible and easy-to-use communication and sensor systems to support post-disaster management (i.e. the organization of actions in the follow-up of disasters), especially when critical infrastructure is affected. This paper introduces a system design that combines mobile XMPP-based and sensor-equipped devices with the flexibility of cloud services. This combination provides the communication between the different involved parties (e.g. rescue forces) 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.**

***Index Terms*—XMPP, Cloud Services, Collaboration, mDNS**

I. Introduction

Natural threats like earthquakes, tsunamis, floodwaters, tornadoes, and fires are usually hard to predict. An accurate disaster management in the follow-up of natural disasters is fundamental to enable a quick evaluation of the critical situation as well as the safeguarding of infrastructures and human lives. If the disaster management team has no overview of the incident or the current situation, correct decisions and quick actions are hard to take, as the series of accidents in the Fukushima nuclear power plant, following the earthquake and tsunami catastrophe in Japan, has shown. Without access to necessary information, rescue teams and emergency forces are often in doubt about the best way to handle a critical situation. It is therefore essential to support post-disaster management and rescue forces with a quickly deployable and easy-to-use system that supports communication and collaboration of the 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 monitoring system of cooperating XMPP-driven entities with data gathering and analysis support through collaborative cloud services.

Flexibility, reliability, cost-efficiency, and a wide applicability of the monitoring system are our main concerns. The openly standardized *Extensible Messaging and Presence Protocol* (XMPP) [1], [2] is therefore used as the basic communication protocol. XMPP is a set of flexible and open XML technologies standardized by the IETF and widely used by companies like Google [3] or Facebook [4] in various appliances. Through its decentralized client / server architecture, XMPP is fault tolerant and minimizes the single point of failure problem. Deployed in applications running on small and mobile hand-held devices equipped with exchangeable sensors, XMPP allows communication partners to interact and communicate as well as to collect data from sensors and share it with other devices or users. The applications and the underlying technology should hence 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 deploying it into a cloud-based data storage pool [5] for further analysis and processing with cloud services. Various cloud-based services are imaginable here, e.g.: plain web browser access of sensed data, simplification of decision processes through ad hoc cooperation of rescue teams, posterior data analysis through statistical processing inside the cloud.

An important aspect of XMPP is the possibility to interact with a server infrastructure (XMPP Core) as well as the alternative of ad hoc communication (P2P) by using the *XMPP extension* (XEP) *XEP-0174 Serverless Messaging* [6]. XEPs [7] in general extend XMPP with additional capabilities. XEP-0174 enables ad hoc communication through the use of *DNS Service Discovery* (DNS-SD), *Multicast DNS* (mDNS) and *Zeroconf* [8]. 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 scenarios, in contrast to infrastructure scenarios where the central XMPP server provides unique *Jabber IDs* (JIDs). We address this problem by introducing an XMPP-driven data collecting system, designed especially for hybrid network environments, where data needs to be located both in ad hoc as well as infrastructure scenarios with support for unique JIDs.

This work provides two contributions: It presents the system design with an analysis and initial evaluation of its underlying networking aspects (e.g., switching between infrastructure and ad hoc networks, providing unique JIDs). With the evaluation, new insights on the practical appliance of XMPP for collaborative applications are gained and presented. We also show that the requirements to assist disaster management are met by our XMPP- and sensor-equipped hand-held devices in combination with the flexible cloud-based storage and processing services.

The paper is organized as follows: Section II introduces our scenario and the corresponding requirements. The state-of-the-art is surveyed in Section III with a discussion of related work in Subsection III-C accordingly. The system design is introduced in Section IV, together with a discussion on the current implementation state and possible future developments. Section V summarizes our initial evaluations of the prototypical implementation while Section VI provides concluding remarks and an outlook on future work.

II. Scenario Considerations

Crisis and post-disaster management is never a fixed nor predetermined work. Every crisis has its particularities and approaches that might work at a certain place, time, or stage might not work at a different one. It is hence important to gain as much information as possible about the current situation and the status of affected people, buildings, landscapes, and infrastructures. Only then can rescue forces take the best possible steps according to the situation.

Nowadays, early warning and sensor systems are often deployed to cope with natural or man-made 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 have shown that pre-planned precautions are often not sufficient and deployed sensors and communication infrastructure get destroyed, hence becoming unusable by rescue forces. Figure 1 depicts such a case where a flood destroyed the communication infrastructure and the status of the critical infrastructure (power plant) is unknown to the rescue forces. Examples like this require a post-disaster sensor and monitoring system that needs to be easily deployable and flexible to support crisis management and rescue forces in every possible way.

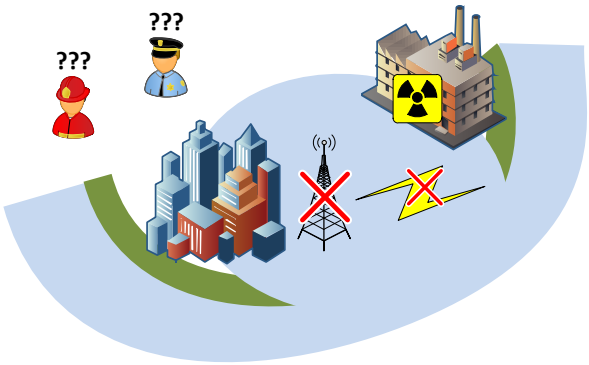

Fig. 1. Example Scenario

This 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 system design and solution approach:

- Supporting different communication parties / partners;
- Providing global view on self- and remotely-sensed data;
- Allowing access for locally and remotely involved parties;
- Using open and standardized communication protocols;
- Enable data access with widely available applications;
- Providing data storage and posterior data analysis capabilities through cloud-driven services;
- Ease-of-use and flexibility of the deployed solutions.

The system design itself is described in detail in Section IV. In general, it is designed for a shorter working period (weeks up to months), in contrast to typical wired or wireless sensor network deployments for monitoring or disaster alerts [9], which are supposed to run for long time periods (years). We use standardized communication technologies like WLAN or WWAN (e.g., GPRS, UMTS) 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 situation reports and in making adequate real-time situation-dependent choices. A further combination of sensed data, place and time of the sensing, and the specific sensing entity (e.g., firemen, rescue specialist) enables an even wider range of applications, from environmental monitoring to marking of dangerous spots or traceable scanning of sites for endangered civilians.

III. Technical Background and Related Work

This section summarizes the state-of-the-art of technologies and protocols that we use in our system. It concludes with a discussion of and separation from related 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. 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 hence interconnected by a domain-based network of servers. A single point of failure or overload situation 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 underlaying communication protocol is that the XMPP Standards Foundation offers an open, standardized, and continuous maintenance for the protocol family, which allows system designers to benefit from the aspects of sustainability. Furthermore, XMPP offers a rich variety of open source software for servers, clients, and libraries [10] supporting several operating systems, thus reducing the costs for development and testing. A diversity of different systems, ranging from desktop computers to mobile entities, can easily be connected with XMPP. It supports various application types besides the usual messaging or presence propagation. Examples can be found in the field of ad hoc grid computing [11], intercloud directory and exchange [12], or semantic monitoring [13].

Another important aspect is that XMPP using *XEP-0174 Serverless Messaging* enables communication over LANs or WANs without need for server infrastructure. This method uses the zero-configuration networking principle (cp. Section III-A) to discover entities that support the protocol and then to negotiate a serverless connection to exchange XMPP messages.

A. mDNS & DNS-SD

Multicast DNS (mDNS) is based on the Domain Name System (DNS) protocol [14]. 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 address `224.0.0.251` (IPv4) and `ff02::fb` (IPv6) and the UDP port `5353` are likewise reserved by the IANA [15]. DNS Service Discovery (DNS-SD) enables to locate and to publish services in 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 when devices want to know the hostname of a node given by its network address. In mDNS it is used to assign service instances to a service.
- *TXT:* to propagate user-defined text, e.g., distribute presence in XMPP *Serverless Messaging* (cp. Section IV-C1).

B. Cloud Services and Storage

Nowadays cloud services give any user the chance to access a personalized, highly scalable, reliable, secure, and fast infrastructure hosted by a third party. Several hosting providers, like Amazon [16], IBM [17], and Microsoft [18] operate large data centers offering a wide range of cloud service products for moderate prices. Charge calculation is done by the use of these services for the amount of storage and the corresponding data transfer amount. 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 in order to provide large amounts of online storage to support collaborative work without the need to install physical storage devices in the user’s datacenters or offices. 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 [19].

C. Related Work

Combining sensors with cloud technology is a new trend as explained in [20], [21]. A main advantage is that real-time sensor data can be processed instantaneously, but using a Wireless Sensor Network (WSN) for this has several drawbacks [22]: Energy is a big concern for resource constrained sensor nodes, so data needs to be transported hop-by-hop to a network sink, causing delays in real-time data streams. In addition only small data packets are usually sent through the WSN, limiting the level of detail during the data analysis (cp. 127 Bytes max. 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.

Our system design omits using WSNs. A sensor device in our design is connected either directly to the Internet or through a one hop communication via XMPP to deliver a real-time data stream with a high resolution directly to the cloud service. Querying data from such devices at close range can be done through *XEP-0174*. In contrast to WSNs, where gateways are required for interconnection of different protocols or sensor networks, we only use XMPP as the sole communication protocol to offer clients a transparent access without the need for a middleware. This measure could also solve the problem of non-standard software interfaces of sensors, already listed in [21]. Another important aspect separates our system from WSNs: Our system is designed for a shorter life-cycle, which is caused by the fact that monitoring critical infrastructures in post-disaster situations is only a temporary challenge. Shorter here means a period of weeks or months, compared to years or longer for structural health monitoring with WSNs [23], [24]. WSNs are also often limited in their sensing capabilities due to their constrained nature. More important then a long life time for us is hence to provide an uncomplicated system with an easy setup and comprehensible data access. Using 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 sensors are used, where to place the sensor nodes, how to access them in-field, or how to interconnect the sensor network to any external network (e.g. the Internet). In contrast, our design shall provide the flexibility to change these decisions alongside and after the deployment, thus separating it from WSNs.

IV. 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 cloud services 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 XMPP networks through unique Jabber IDs (JIDs). This elementary part is described below in detail, while an evaluation with a mobile hand-held prototype is outlined in Section V.

A. Architecture of the System

The system consists of sensor-equipped devices (not to confuse with wireless sensor nodes) and the cloud service as depicted in Figure 2. Each sensor device runs an XMPP software client through which it can publish sensed data. The cloud service manages the storage pool for the measured sensor data as well as several XMPP domains which allow an interconnection and data exchange between different organizations (e.g., rescue team, government, private organization).

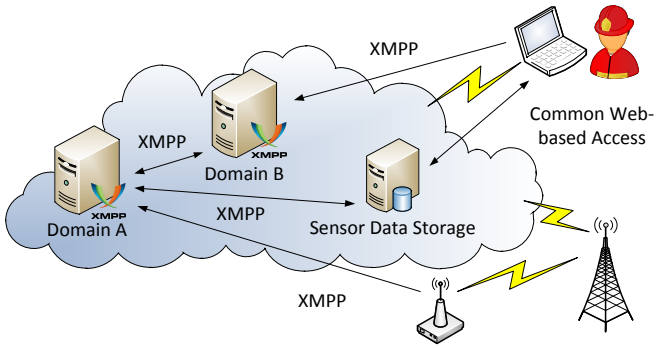

Fig. 2. System reference model

B. System Criteria

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

a) Standard protocols: XMPP and mDNS are established standards for detection of entities in ad hoc networks and for collaborative communication. In contrast to WSNs, where gateways are needed to couple 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 [25] for the event distribution.

b) Temporary use: Emergency situations occur over a limited time period. Thus our proposed system is designed to support runtime durations up to several weeks or months (cp. Section III-C). 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.

c) Transportable entities: Sensor-equipped devices should be compact and transportable. We favor the use of embedded systems running Linux instead of WSNs with proprietary operating systems to reduce maintenance and development costs. In the broader sense it offers us the possibility to run standard implementations of mDNS [8], [26] and XMPP on such devices.

d) Sensor groups: Sensor devices may consist of several detectors (e.g., compression, humidity, temperature, acceleration, etc.). Grouping of detectors will help handling a large number of theses 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 be easily managed in a single sensor group.

e) Seamless integration: Integrating our system seamlessly 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 stressful situations. Therefore sensor data access can be realized via a standard XMPP chat client, ensuring a slim and fast implementation for several devices on various operating system, ranging from desktop systems to hand-held devices.

f) 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, since no special hardware must be developed. Through the use of existing XMPP libraries, server and client development costs and test periods can be reduced, because designing, implementing, or intensively testing a new underlying network protocol can be skipped.

g) Automatic configuration: User-driven configuration of the system should not be required. Automatic configuration shall be an integral system part, 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 autonomously.

h) Network access: The system should support a wide range of network access technologies. A direct access to the cloud 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) or Bluetooth should be used to share the Internet access throughout the XMPP network.

i) 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 hence.

j) 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 the understanding of the received data. The whole event itself is logged in the data history and can be used as a review or for overlaying functions for maps for example.

k) Data exchange: During an emergency situation 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 both to the sensor devices or to the cloud service. Only a simple XMPP client is needed, which can be installed ad hoc for several platforms.

C. 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 in an ad hoc or in a infrastructure network. While entities with Internet access may connect directly to an XMPP server, user in ad hoc networks without Internet sharing may find entities through the XMPP Extension Protocol (XEP) *XEP-0174 Serverless Messaging* [7]. For coupling ad hoc networks with the conventional XMPP infrastructure an agent is used. 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 of 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 [27]), thus eliminating the need for a system specific user identifier. Figure 3 illustrates the network structure of the localization system.

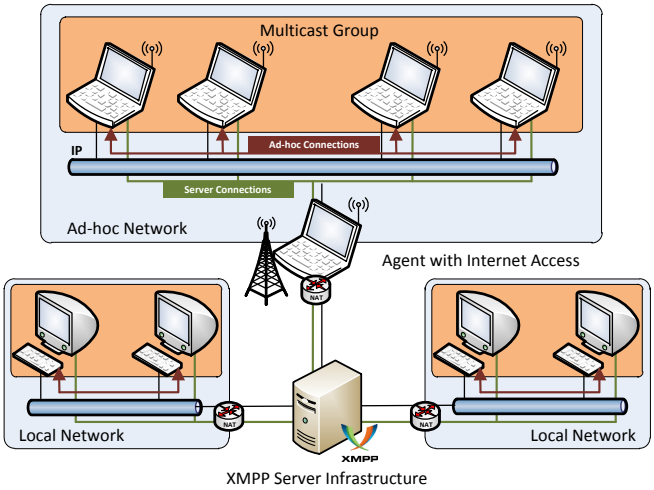

Fig. 3. XMPP network structure

1) XMPP Serverless Messaging: The mobile entities form an ad hoc network. After this they join a multicast group on address `224.0.0.251` for IPv4 and address `ff02::fb` for IPv6. A user advertises his 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. 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 user information (one SRV, TXT, A, and PTR record per user). If a user wants to leave the network, he has to send a PTR record with TTL set to zero. Each node analyses the released records inside the multicast group and each of them generates a list of entities based on the received records.

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) [28] protocol. Normally, SRP was designed for client / server authentication. Modifications were necessary to use SRP for pairwise authentication in ad hoc networks. During the authentication step a shared password between both nodes is used. The exchange of transport addresses is realized over the authenticated and encrypted communication channel by *Jingle*.

D. Access Strategies

The access of 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, a sensor-equipped device, the sensor data storage, and the described access strategies.

1) Data Collecting: Our sensor devices can be equipped with any available detector to enable data collecting, depending on the requirements of the current situation. 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 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 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 [29]. 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)*.

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 todays 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 mathematic operations *AND*, *OR*, and *NOT*.

3) Communication Flow: Figure 4 depicts the detailed communication flow with the following steps:

I: Devices start connecting to the Internet (I.1) and register themselves in an XMPP domain (I.2);
II: Sensor devices join a virtual group, depending on their GPS position and attached detectors;
III: Devices start collecting data and send it to the Internet;
IV: Fall-back via *XEP-0174*: devices can be accessed directly by the rescue workers in the outer quarter.

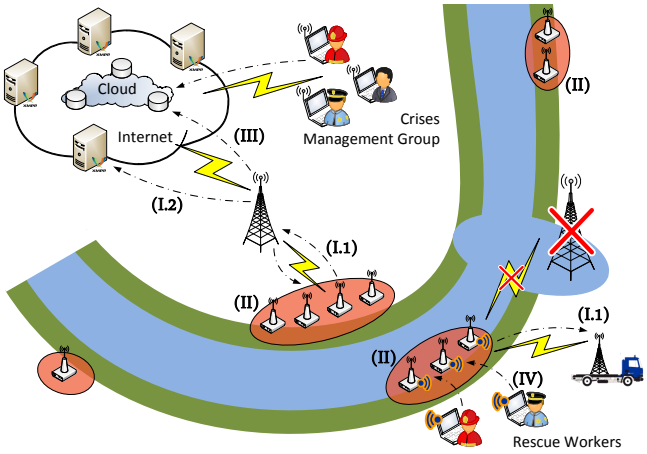

Fig. 4. System communication flow

The fall-back mechanism of our system is based on *XEP-0174 Serverless Messaging* [7]. 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, or for areas where radio transfer is short-staffed.

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

A sharing or joint-use sensor device module builds up a group of many sensor devices to monitor a detailed area. All sensor 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 devices while the access to the measured sensor data is still transparent through XMPP.

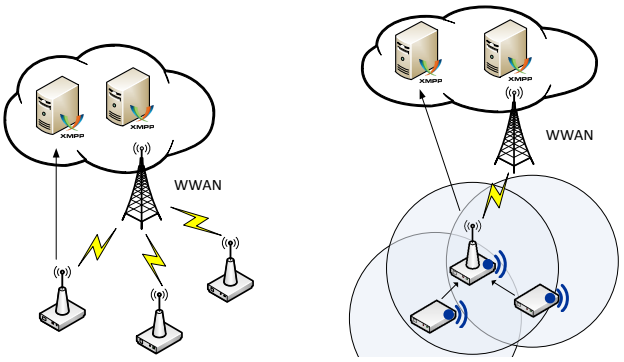

Fig. 5. Dedicated (a) and Shared (b) sensor device module

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. The cloud consists of the sensor data storage and an analysis tool for the collected sensor data, optionally a dedicated XMPP server. Normally our system makes use of public XMPP servers [30] because they are available in manifold configurations, well maintained, and free in usage. Under certain circumstances it can make sense to deploy a private XMPP server, for which the preconfigured XMPP component is ready to be used. The sensor data storage acts as a normal XMPP client inside the network and thus 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 service and 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 hour. The whole analysis component can be frozen if not needed and started again on demand.

Through the collaboration of the sensor devices with the cloud service, a set of new 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 sensed data can be viewed from each control point along with current measured data;

- Conspicuous 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 workers in the outer quarter via *XEP-0174*;
- Sensor device modules from different organizations can be interconnected through XMPP, while providing a continuous and transparent view on the sensed data.

Results generated by the 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 current sensor data directly, or a feature rich web view applied by the cloud service with prepared sensor data.

5) Collaboration: 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 the collected data between the crisis management and the rescue forces. In collaboration new tasks for each rescue team can be adjusted ad hoc and managed by the crisis management as a joint task. Communication groups and chat rooms via *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-workers, 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.

E. Sensor Device Prototype

Our prototype is based on an HTC HD2 smartphone [31]. 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. It is therefore a good example for running our XMPP localization and submitting the currently measured sensor data to the Internet. Furthermore it meets our system criteria introduced in Subsection IV-B

1) Application Scenario: Depending on the described scenario in Section II the sensor devices can be equipped with several detectors such as compression, humidity, radiation, or acceleration. In our application scenario the smartphone can be used to detect dynamic loads on buildings while sending the measured data from the accelerator to the Internet. Therefore the device could simply be attached to the walls of a building.

2) Used Libraries: This work resorts to standardized technologies (mDNS, DNS-SD, XMPP, TLS, SRP), available for various programming languages, platforms, and systems. The criteria for choosing libraries were: supporting a wide range of operating systems (Windows Desktop/Mobile, Linux, etc.) and platforms (x86, ARM, etc.); written in `C/C++`; and available under a free software license. We assume that there is at least one `C/C++`-compiler available on each platform. The libraries `mDNSResponder` [26], `gloox` [32], and `OpenSSL` [33] were chosen based on the mentioned criteria. mDNSResponder implements the mDNS- and DNS-SD-protocol. A client side implementation of the XMPP Core is provided by gloox while OpenSSL implements the TLS / SSL protocol.

V. Performance Analysis

This section analyzes the performance of the systems fallback 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 and vice versa. We can hence find out how fast our devices switch between both networks. 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 the scaling performance of our system 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 simulate the application scenario in Section IV-E1. This minimal 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 was very short and cost effective, because existing libraries could be reused for all involved devices as described in Section IV-E2. We chose the XMPP server from the *xmppnet project* [34] for the test, 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 WLAN-driven 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 which occur by handling further XMPP messages to the XMPP server and vice versa.

A. Test Case 1: Node left the Ad Hoc Network

In test case 1 node A left the ad hoc network and created an 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 6 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 6. 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 $1s$. 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 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 total network switching time. These delays are established at the server by the processing of presence information as well as in the test application itself. Furthermore 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 lie in the test application itself, like the delayed notification of currently activated UMTS interfaces, so that an earlier initiation of a server connection might not be possible.

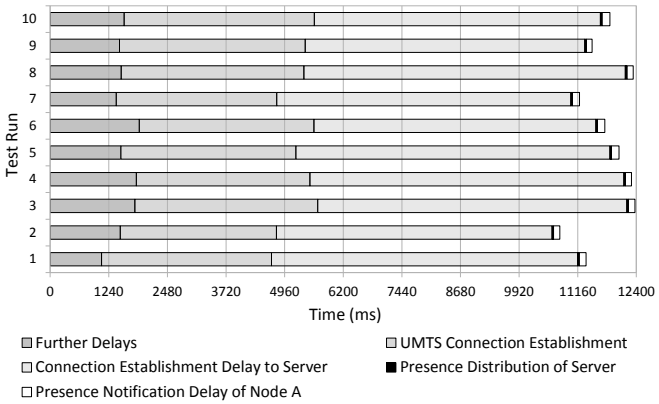

Fig. 6. Results of *Test Case 1* - Node left the ad hoc network

B. 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 7. 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 takes 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 if 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 $2\ s$ (cp. [35]). 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.

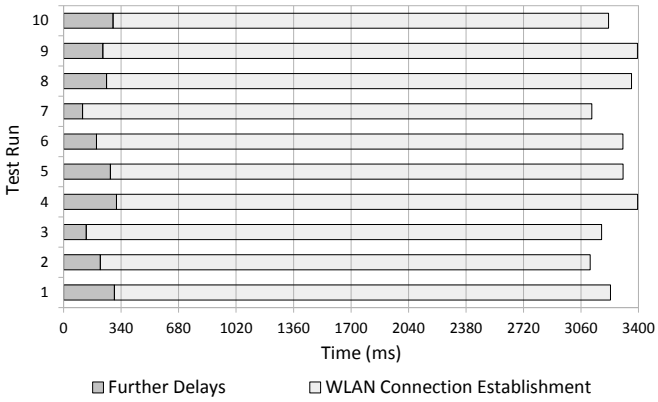

Fig. 7. Results of *Test Case 2* - Node joined the ad hoc network

C. 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 I. The protocol layers (DNS, UDP, IP) below DNS-SD use `40` Bytes constantly for every message.

TABLE I
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 join 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 KB have to be transferred in the worst case. This amount of data traffic is acceptable according to our evaluations. We further used compression methods to reduce this data traffic. Two methods are available: the *Known-Answer Suppression* [36, Sec. 7.1] and the *Duplicate Question Suppression* [36, Sec. 7.3] method. Both methods 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 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 example 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 (cp. Section VI).

D. Result of the Performance Analysis

In conclusion we can say that the XMPP implementation can be used to locate users by their unique JID 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. Furthermore an estimation of the scaling performance of mDNS for the maximum number of participants in the ad hoc network was given.

VI. Conclusions

This paper presented an XMPP-driven system combining sensor equipped devices with cloud services. We showed that different 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 WSNs. Through the implementation and evaluation of a sensor device prototype, 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. We also described the integration of the cloud system as a data storage and data aggregation center that is scalable and ensures the reliability of sensed data.

Further work in this area resides in the automatic configuration of XMPP to add and connect new entities of possible foreign organizations autonomously. Normally a new entity with a predefined JID needs to register itself at the domain of the XMPP server. It subsequently grants access to its presence by answering subscription requests from other entities. Only then can other entities monitor the presence of the newly joined entity. This process needs to be done in the background at the first power-on and subsequent initialization of a sensor device and should progress without human interaction. Investigations in this field could be done based on an initial message with encoded access parameters or the plain use of pre-configured devices. The advantage of using access parameters hashed as a key to configure the XMPP client is that joining observers gain easy access to the measured data of all sensor devices by receiving an XMPP message without the need to know which entities are already in the XMPP network. The drawback of this approach is that a new message type is necessary, which will cause an interoperability loss with all current XMPP chat clients. A grouping of entities could alternatively be performed during the bootstrap phase through a pre-configuration of devices. Entities would join a default bootstrap group based on *XEP-0045 Multi-User Chat*. Then the group could be accessed by joining entities using available XMPP libraries and be observed by using available XMPP chat clients.

Moreover, we are going to perform simulations of mDNS to validate its scaling in large hybrid networks. This can help to identify bottlenecks or possible optimizations in the code of existing implementations as mentioned in Section V-C. To support the field of tiny, embedded hardware, the mDNS and XMPP implementations need to be extremely lightweight, because resource constrained embedded systems have small memory sizes and slow microcontrollers. The idea here is to optimize XMPP for the use in constrained devices like it is currently researched for the Simple Network Management Protocol (SNMP) [37]. The project uXMPP [38], a lightweight implementation of the XMPP protocol, could be used as a starting point for research and acceptability testing. A closer comparison of WSNs and our XMPP-driven hybrid network could be done in the future to further analyze possible advantages of publish/subscribe based XMPP networks over polling and routing based wireless sensor networks.

Sensor devices are designed to support different use cases and scenarios, as mentioned earlier. Each device can be configured with an individual set of sensors and a sensing

threshold. Programming a large number of devices is time consuming work. For this reason, we want to develop a remote update routine based on the flexible P2P group communication protocol *Moversight* [39], which is specifically designed for mobile devices and hybrid network environments. Moversight provides a virtual and synchronous data transport service among group members. Using this service as a lightweight programming tool we will be able to provide a remote update service for a group of sensor devices, which can be used to upload new configurations or new system images to each device in the background while running components will be unaffected. Consequently, using this update method will guarantee an identical configuration and system version for a sensor device group at the system start-up or at a reboot, thus avoiding misconfiguration issues or the use of incompatible software versions during runtime. Compared to the Network Configuration Protocol (NETCONF) [40], which is designed to manage single devices in a network, development and test costs can hopefully be reduced, because interoperability issues will be avoided. The main focus of our Moversight-based remote update routine is to allow software and configuration updates for a group of devices instead of a single network device and to ensure that all group members successfully complete the update, thus enabling a better runtime maintenance for all kinds of post-disaster management applications.

REFERENCES

[1] P. Saint-Andre, “Extensible Messaging and Presence Protocol (XMPP): Core,” IETF, http://www.ietf.org/rfc/rfc6120.txt, Request for Comment 6120, Mar. 2011.

[2] “XMPP Standards Foundation,” http://xmpp.org/, 2011.

[3] “Google App Engine, The XMPP Java API,” http://code.google.com/appengine/docs/java/xmpp/, 2011.

[4] “Facebook Chat Now Available Everywhere,” http://blog.facebook.com/blog.php?post=297991732130, 2011.

[5] J. Peng, X. Zhang, Z. Lei, B. Zhang, W. Zhang, and Q. Li, “Comparison of Several Cloud Computing Platforms,” *Second International Symposium on Information Science and Engineering*, 2009.

[6] P. Saint-Andre, “XEP-0174: Serverless Messaging,” XMPP Standards Foundation, http://xmpp.org/extensions/xep-0174.html, Standards Track, Nov. 2008.

[7] “XMPP Extensions,” http://xmpp.org/xmpp-protocols/xmpp-extensions/, 2011.

[8] The Avahi Team, “More About Avahi - Details about mDNS, DS-DNS and Zeroconf,” http://avahi.org/wiki/AboutAvahi, 2011.

[9] P. Morreale, F. Qi, and P. Croft, “A Green Wireless Sensor Network for Environmental Monitoring and Risk Identification,” *International Journal of Sensor Networks 2011*, vol. 10, no. 1/2, pp. 73–82, 2011.

[10] “XMPP Software,” http://xmpp.org/xmpp-software/, 2011.

[11] G. Weis and A. Lewis, “Using XMPP for ad-hoc grid computing - an application example using parallel ant colony optimization,” *IPDPS ’09 Proceedings of the 2009 IEEE International Symposium on Parallel and Distributed Processing*, 2009.

[12] D. Bernstein and D. Vij, “Intercloud Directory and Exchange Protocol Detail using XMPP and RDF,” *IEEE 6th World Congress on Services*, 2010.

[13] D. Renzel and R. Klamma, “Semantic Monitoring and Analyzing Context-aware Collaborative Multimedia Services,” *Proc. of the 2009 IEEE International Conference on Semantic Computing*, 2009.

[14] P. Mockapetris, “Domain Names - Implementation and Specification,” IETF, http://www.ietf.org/rfc/rfc1035.txt, Request for Comment 1035, Nov. 1987.

[15] IANA, “Port Numbers,” http://www.iana.org/assignments/port-numbers, 2009.

[16] Amazon, “Web Services,” http://aws.amazon.com/, 2011.

[17] IBM, “Cloud computing,” http://www.ibm.com/cloud-computing/us/en/, 2011.

[18] Microsoft, “Windows Azure,” http://www.microsoft.com/windowsazure/, 2011.

[19] Y. Wang, X. Wen, Y. Sun, Z. Zhao, and T. Yang, “The Content Delivery Network System based on Cloud Storage,” *Proceedings of the International Conference on Network Computing and Information Security*, May 2011.

[20] V. Rajesh, J. M. Gnanasekar, R. S. Ponmagal, and P. Anbalagan, “Integration of Wireless Sensor Network with Cloud,” *International Conference on Recent Trends in Information, Telecommunication and Computing*, 2010.

[21] L. Shu, M. Hauswirth, L. Cheng, J. Ma, V. Reynolds, and L. Zhang, “Sharing Worldwide Sensor Network,” *International Symposium on Applications and the Internet*, 2008.

[22] J. Heidemann, F. Silva, C. Intanagonwiwat, R. Govindan, D. Estrin, and D. Ganesan, “Building Efficient Wireless Sensor Networks with Low-Level Naming,” *Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP ’01)*, Oct. 2001.

[23] S. Kim, S. Pakzad, D. E. Culler, J. Demmel, G. Fenves, S. Glaser, and M. Turon, “Health Monitoring of Civil Infrastructures Using Wireless Sensor Networks,” EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2006-121, Oct 2006. [Online]. Available: \url{http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-121.html}

[24] N. Xu, S. Rangwala, and et al, “A Wireless Sensor Network For Structural Monitoring,” in *Proceedings of the ACM Conference on Embedded Networked Sensor Systems*. ACM Press, Nov. 2004, pp. 13–24.

[25] T. Eugster, A. Felber, R. Guerraoui, and A.-M. Kermarrec, “The many faces of publish/subscribe,” *ACM Comput. Surv. 35 (2)*, 2003.

[26] Apple Inc., “mDNSResponder,” http://www.opensource.apple.com/tarballs/mDNSResponder/, 2011.

[27] Google Inc., “Google Talk,” http://www.google.com/talk/, 2011.

[28] T. Wu, “he SRP Authentication and Key Exchange System,” IETF, http://www.ietf.org/rfc/rfc2945.txt, Request for Comment 2945, Sep. 2000.

[29] “Pidgin Development Wiki - Supported XEPs,” http://developer.pidgin.im/wiki/SupportedXEPs, 2011.

[30] “Free XMPP Server List,” http://www.jabberes.org/servers/servers_by_times_online.html, 2011.

[31] HTC America Inc., “HTC HD2 Specifications,” http://www.htc.com/us/products/t-mobile-hd2#tech-specs, 2011.

[32] J. Schroeter, “Gloox,” http://camaya.net/gloox/, 2011.

[33] OpenSSL Development Team, “OpenSSL Website,” http://www.openssl.org/, 2011.

[34] i-pobox.net, “XMPPNet Project,” http://www.i-pobox.net/, 2011.

[35] S. Cheshire, B. Aboba, and E. Guttman, “Dynamic Configuration of IPv4 Link-Local Addresses,” IETF, http://www.ietf.org/rfc/rfc3927.txt, Request for Comment 3927, May 2005.

[36] S. Cheshire and M. Krochmal, “Multicast DNS,” IETF, Internet-Draft, Feb. 2011.

[37] J. Schoenwaelder, H. Mukhtar, S. Joo, and K. Kim, “SNMP Optimizations for Constrained Devices,” IETF, Internet-Draft, Oct. 2010.

[38] E. Bail and A. Hornsby, “uXMPP implementation for Contiki Operating System,” http://www.cs.tut.fi/~hornsby/research.html, 2008.

[39] J. Gaebler and M. Kirsche, “Moversight - A Flexible P2P Group Communication Protocol for Mobile Collaborative Applications,” *18th IEEE International Conference on Network Protocols (ICNP 2010) - Poster Session*, Oct. 2010.

[40] R. Enns, M. Bjorklund, J. Schoenwaelder, and A. Bierman, “Network Configuration Protocol (NETCONF),” IETF, http://www.ietf.org/rfc/rfc6241.txt, Request for Comment 6241, June 2011.