DICE: A Novel Platform to Support Massively Distributed Clouds

Anna Förster[1], Koojana Kuladinithi[2],
Andreas Timm-Giel[3], Carmelita Görg[2], and Silvia Giordano[1]

[1] University of Applied Sciences of Southern Switzerland
[2] University of Bremen, Germany
[3] Hamburg University of Technology

Abstract. Massively distributed clouds (MDC) have a huge potential in serving novel applications and services in many situations. Mainly, they are able to provide communication without the use of an infrastructure and to guarantee full data and user anonymity. However, their implementation is not trivial and requires innovation in many different fields, such as opportunistic communications, big data management and security. In this paper, we present our first design of a MDC supporting architecture, called DICE: Distributed Infrastructureless Cloud sErvices. We present our main application scenario and focus on implementation challenges and early results.

1 Introduction

Recently, the idea of massively distributed clouds (MDC) has emerged, where services are provided in a completely distributed and infrastructureless manner. Many applications, e.g. environmental monitoring, participatory sensing, or social networking, can be much better served by a such a paradigm, where users do not need to use or pay for an infrastructure. Instead, they can leverage the direct communication between devices in their proximity to exchange localized data.

The main novel properties of MDC are a decentralised architecture with infrastructureless communications and data anonymity, which would indeed change the existing understanding of cloud services and their markets. Its realisation relies on innovation in opportunistic and heterogeneous communications, security and privacy, big data management, and energy management. These make the implementation of an MDC a challenging task. Furthermore, real users and their behaviour need to be incorporated very really in the implementation process, as they cannot be simulated satisfactorily.

On the other side, implementation of MDCs has been hindered also by prejudices in the area of wireless communications, where the prevailing opinion is that only a stable and highly available infrastructure can guarantee a particular level of service. However, even the best infrastructure cannot guarantee the timely and local availability of data itself, such as touristic information or environmental properties. Such data is mostly gathered by end users and needs to be

D. Pesch et al. (Eds.): MONAMI 2013, LNICST 125, pp. 273–286, 2013.

inserted by them into the data cloud to be useful. However, current practice does not allow to do this in a cost-free and user-friendly manner, thus data quality suffers. On the other hand, opportunistic communications require people and devices to cooperatively exchange data and to be geographically close to each other. This is typically seen as a limitation, as it possibly delays the propagation of data. However, our own preliminary study on the impact of device density on data propagation in a purely opportunistic environment [2] has shown that even low country-wise population densities can support fast data propagation.

Motivated by these early results and by the high interest shown in the community [1,2] we have decided to proceed with a prototype implementation of an architecture, able to support MDCs, called DICE (Distributed Infrastructureless Cloud sErvices). In the rest of this paper, we first present and discuss some related efforts in Section 2, before presenting our main application scenario in Section 3. Section 4 gives an overview of the DICE architecture, which is under active development. Then we discuss possible usage and business models for our envisioned services in Section 5. Finally, we summarise our next steps and conclude the paper in Section 6.

2 Motivation and Related Efforts

The motivation of our work is many-fold. It is mainly based on the general user dissatisfaction with existing cloud and mobile services, their usability, functionality and privacy preservation. There are three main arguments, which we discuss here: *service accessibility*, *service usability* and *user privacy*.

2.1 Service Accessibility

Currently cloud services are in theory accessible by anyone at zero cost. However, reality looks differently. A clear requirement to use any of these services is to have Internet connection. However, Internet penetration rates worldwide were still only 34.3% in 2012[3]. Even in the most "connected" continent, North America, the penetration rate was 78.6% – thus, at the best case, only two third of the population has access to these services. In the worst case, in Africa, only 15.6% can leverage them.

We can already see some interesting approaches of providing Internet connection to remote areas, such as Google's Loon Project[4]. The main idea is to launch Internet Balloons over rural areas, which connect to satellites to provide Internet connection. However, such projects are not cost-efficient and do not scale well.

Another challenge are the services themselves. Metropolitan areas tend to have more and more sophisticated services for their citizens than rural areas. Low-population regions are simply not attractive from the business point of view.

[1] `http://www.smartsantander.eu`
[2] `http://citi-sense.nilu.no`
[3] Source: `www.internetworldstats.com/stats.htm`
[4] `http://www.google.com/loon/`

Thus, the gap between urban and rural areas becomes even larger, sharpening also other problems such as health, employment, etc.

Moreover, provision of cloud services seems to be restricted mostly to large industrial players. Independent developers, startup companies, charity organisations and academia have little chances to provide services because of their high support costs. Standard support for cloud services, such as identification and security applications, data management, etc, are costly. This problem has been already partially addressed by the OpenStack initiative[5], which provides such services to developers free of charge. While this is an important step, the service and infrastructural support remain a stumbling block.

2.2 Service Usability

Existing cloud and mobile applications and services typically concentrate on one particular application scenario, such as place rating, message exchange, data storage, etc. Some of the most prominent examples are listed and compared in Table 1, such as Facebook, Twitter, Foursquare, Dropbox, etc. A very simple example is provided by state-of-the-art everyday communications and the "share" buttons, e.g. to share something on Facebook, Twitter or Wordpress. There, sharing a particular piece of information is simple and at the same time complex: it is simple to push the button, but then you need to decide which of all available services to use, maybe you need to authenticate yourself again, to write a small comment, etc. This broad spectrum of various social networks is

Table 1. Comparison between existing web and cloud services and the envisioned Massively Distributed Heterogeneous Clouds.

Service/ Application	Network Connectivity		Architecture		Communication			Anonymity /Privacy	Geographic Relevance		Usage	
	Infra-structure	No Infr.	Centr.	Distr.	Single User	Mult. Users	Any User		Global	Local	Passive	Active
Social networks (Twitter, Facebook)	X		X		(X)	X	(X)	none	X		(X)	X
News Feed (RSS)	X		X			X		hidden from other users	X		X	
Streaming (Youtube)	X		X			(X)	X	hidden from other users	X		X	
Community-based webpages (Tripadvisor, Foursquare)	X		X				X	partial		X	X	X
Event or Tourist Information (LongLake Festival Lugano)	X		X				X	hidden from other users		X	X	
Participatory Sensing (Bikenet)	X		X				X	hidden from other users		X		X
Cloud Storage (Dropbox)	X		X		X	X		none	X			X
Cloud Services (Googledocs)	X		X		X	X		none	X			X
MDC (DICE)		X		X			X	anonymous	X	X	X	X

[5] http://www.openstack.org

a very good example of the rather bad usability of current mobile services and their high complexity.

Another example is the broad spectrum of locally available services and applications. Nowadays, every festival and every event in any city provides its users with its own mobile application. However, before such an app becomes useful, it needs to be found and installed. Many disappoint with bad user interface, with missing language settings or with outdated or missing information. In every city, there is currently a myriad of these services and applications and it is impossible even for locals to find and use them all. For tourists, who would profit most from such services, they are impossible to find. Namely, they face also the language and cost challenges: information about services is mostly defined in the local language and the cost for searching for them online is very high because of roaming charges.

2.3 User Privacy

Another challenge is user privacy in these services. Here, we differentiate between *passive* and *active* usage of services. If an application or service has a purely informative characteristic, like an event or tourist service, the user is *passive*, as she never inserts any information. However, even if the user is only asked to rank or review a particular item, like a hotel or a restaurant, she becomes an *active* user. In the first case, user privacy or anonymity does not play a significant role. However, also here the complete privacy of the user is violated, as it is known that she uses a particular application. In the second case, user privacy has a significant role, as she gives away personal data.

In our Table 1, none of the presented services or application offer complete data or user privacy. In all cases, the privacy is supported by state-of-the-art authentication and encoding mechanisms, but these do not *guarantee* that the data cannot be viewed by a third party, legally or illegally.

The only solution to this problem is to offer complete anonymity to the users. *This is different from post-anonymization of their data for presentation purposes, as it requires that their personal data or any description of the data (e.g. location and time of acquisition) is never stored nor propagated through the network.*. To the best of our knowledge, this service is not offered by any current service or application. This can be seen also in Table 1, where we also compare existing services and applications to our novel MDC approach.

3 Application Scenario

There exist already many different applications and systems targeted to environmental monitoring. However, they tend to concentrate on one particular issue or topic: e.g. on air pollution, on restaurant rating, etc. We refer to all these applications as *environmental awareness applications*. These applications refer to any kind of simple or complex data, related to our environment. However, in

our scenario, we do not limit in any way the kind of data users can gather or exchange. Instead, users are completely free to gather any data:

- Environmental properties, such as noise levels, humidity, temperature, air pollution, or crowdness.
- Envirornmental ratings, such as a dark street you should avoid, a romantic place for a picnic, or a great street with many restaurants.
- Problems, such as broken street lamps, or trash in the park.

By using such an application, the users will create and leverage a generally better awareness about their environment. They insert this information into the "cloud" and can retrieve it from there. Depending on their needs or interests, they will be able to view instantaneously important information.

3.1 Requirements Analysis

This application scenario exhibits some important **requirements** in order to be truly useful and safe at the same time:

1. **Data heterogeneity:** The cloud must be able to handle any kind of data, without any restrictions in size or type.
2. **Data anonymity:** In order to guarantee safe exchange of data and thus true user security, the data must be completely anonymous.
3. **Scalability:** The cloud must be able to handle very big data.
4. **Low support and usage cost:** To facilitate usefulness, the cloud service must be free of charge for users and very low cost to providers and supporters.

Especially points 2 and 4 are new to cloud services and generally to mobile applications. While current solutions attempt to guarantee user safety with complex and expensive encryption and identification algorithms, we turn to a new perspective: anonymous data. State of the art security mechanisms have one trivial, but crucial disadvantage: they rely on storing user data somewhere. Thus, at least in theory, the possibility to access this data is always present, be it by mistake or on purpose. This problem is often described also as the "friendly, but curious cloud".

With our requirements of data anonymity, the application will become much more lightweight and will in fact guarantee user safety. Provided that data is never associated with any user, it is impossible to break the privacy of the users, on purpose or mistake.

Another requirement worth discussing is the last one, low support and usage cost. This is typically not or only weakly considered in research-oriented applications. Furthermore, it is usually assumed, that cloud and generally mobile applications are low-cost by default. This is not necessarily true, as practice shows. For example, while an application like FourSquare is free of charge, its usage is very expensive in roaming areas. On the other side, such environmental awareness applications are most useful for people out of their comfort zone, i.e. outside their country of residence. Consequently, the usefulness of our application is only provided when usage and support cost are always close to zero.

3.2 MDC Paradigm

The requirements analysis above clearly shows that this application scenario is perfectly suited for the Massively Distributed Cloud (MDC) paradigm, described in [1] and previous sections. MDC not only fulfils all requirements of our application scenario, but also exceeds them. Mainly and most importantly, it fully supports at the maximum possible level the following three:

- Full data and user anonymity
- Complete infrastructural independence
- Zero usage cost

MDC is completely distributed and each device runs its own instantiation of the MDC platform. The so called "cloud nodes" can be any communication-enabled devices: smartphones, sensor nodes, but also more powerful nodes such as laptops or tablets. Depending on their location and available resources, we differentiate between **mobile** against **static** and **fully functional** against **low capability** cloud nodes. While end users typically carry their devices with them (fully functional mobile nodes), special points of interest can be equipped with fully functional, but static cloud nodes. There is no functional difference between these mobile and static nodes. The difference arises rather from their availability at a particular location, thus providing service guarantee. For example, the city counsel office or the train station might decide to fix a MDC-enabled device to make sure that data is always available at a particular location.

Low capability devices are crucial to MDC and its properties. Current Wireless Sensor Network (WSN) installations are under-utilized, running typically a single, simple data gathering application. While a single node has severely restricted resources, a sensor network can have more significant freely available distributed resources. Thus, individual resource-restricted nodes can act as normal MDC nodes, but can also organize into groups and coordinate to manage the available data together. These sensor networks are typically static, but there are also some mobile examples (e.g. sensors installed on buses or trains).

In the next section we will discuss our architectural design called Distributed Infrastructureless Cloud sErvices (DICE) to support such an environmental awareness application, using the MDC approach.

4 DICE Architecture

Figure 1 offers a high level overview of the envisioned DICE architecture. This highlights our first implementation attempt to realise the MDC paradigm. The DICE architecture differentiates between the main cloud platform with its provided services and applications running on top. In contrast to typical middleware solutions, the main DICE platform can also be run stand-alone and provides the user with a simple interface to select sensor sources and to share them. Thus, users may decide to contribute without leveraging the results themselves. This includes especially sensor nodes, where the sensor nodes themselves are not interested in the data, but provide data and resources to the rest of the cloud.

DICE applications run on top of the platform and provide means of processing various sensor data and representing them in a meaningful way to the user. For example, air pollution data can be represented as a map and combined with pedestrian navigation systems. However, also high-level data can be handled, such as problems in the city, where the problem and the required intervention are represented on a map or on a list.

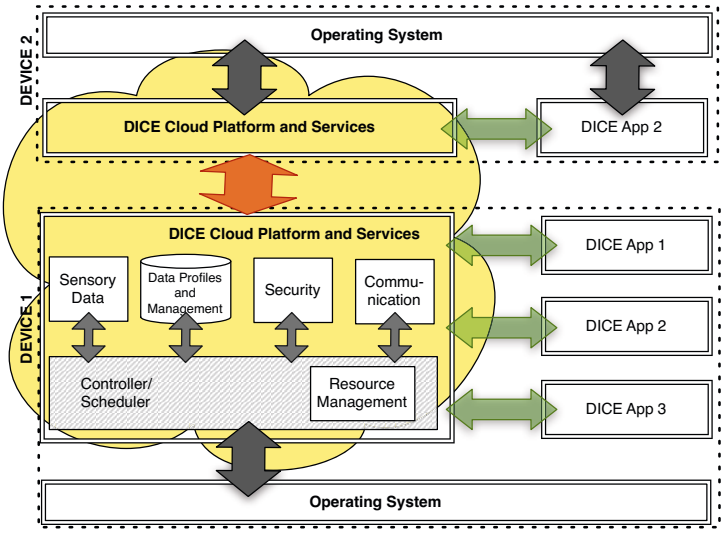

Fig. 1. General DICE Architecture

The main DICE services include data acquisition and management, opportunistic heterogeneous communications, security and resource management. The latter, complemented with a controller and scheduler, builds the central DICE component. It provides resource access to all other services and is crucial to resource-restricted devices such as smartphone or sensor nodes.

4.1 Scheduler and Resource Management

Even with modern smartphones, resource management is an important and crucial task. DICE requires massive data exchange and sensor data acquisition. Thus, a scheduler is used to allow access to different resources and components of the system, such as communication, data storage, sensor data access, etc. The scheduler is combined with a smart resource manager, which learns the behavior of the device (either its user or the applications running on it) to optimally use its free resources.

Therefore, the scheduler monitors and manages the available resources in a very flexible way. It requires a sophisticated interface to all other DICE services, such as communication or data management, and gives them access to resources or hinders them from using resources. An example is newly available data at a

cloud node: the data management service must signal this to the scheduler, who will evaluate the available resources and either allow or disallow the data management to communicate the new data to the cloud neighbors. The functionalities of the scheduler are implemented with state of the art scheduling techniques, complemented with machine-learning techniques for monitoring and prediction of available resources at individual cloud nodes. Figure 2 shows the interaction of the scheduler with other components and services in a fully functional cloud node.

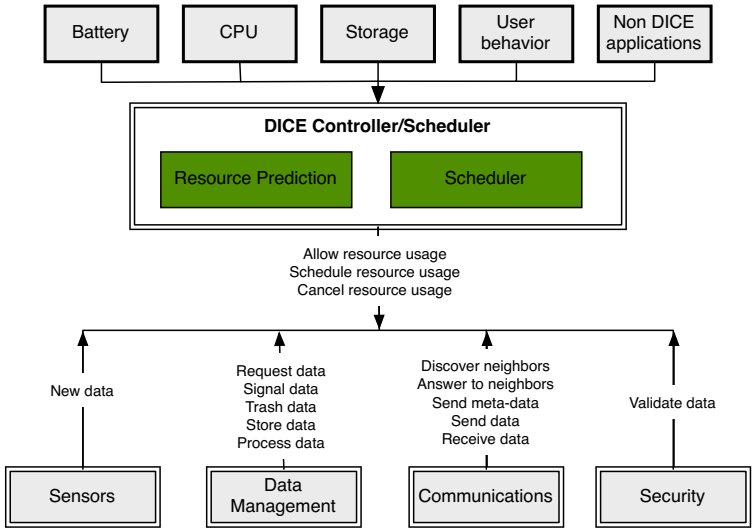

Fig. 2. Overview of the DICE Scheduler/Controller

4.2 Data Acquisition and Management

The data management service of DICE is envisioned to be a highly flexible and smart component, able to identify new data, to aggregate it with existing data, and to identify old or irrelevant data. The main implementation specific requirements are:

- Opportunistic data exchange and management: DICE requires data to be stored throughout the network, and accessible at different points in opportunistic and distributed manner.
- Guarantee the anonymity of data by merging it with existing data and thus loosing any hints to the original source.
- Manage the available resources and compress the available data: For example, while fully functional nodes like smartphones can probably hold a large amount of data, low capability nodes cannot and probably need to organize into clusters to enable meaningful data storage.
- Prepare the data for presentation to the user, e.g. in form of maps or lists of latest data.

In order to fulfill the above requirements in **Data Acquisition and Management** in DICE, we consider the following approaches.

- Use of geo-spatial data stamps and hash-codes: These techniques are used in order to avoid data duplicates while preserving the full anonymity of data. Thus, traditional meta-data, such as creator address, is not necessary to recognise duplicate data.
- Use of smart clustering algorithms: moving the data to the right place where it should be accessed, replicating the data so that it is not lost in case of failure, etc. Furthermore, local organization and role assignment will be considered in order to cope with the resource restrictions on low capability nodes. For example, a single node can take over the role of an access node for several of its neighbors. The same node can also serve as a communication gateway to smartphone and other low capability cloud nodes.
- Novel data distribution and aggregation techniques: Some cloud nodes act as on-the-fly aggregation points (very different from standard aggregation). Therefore, we consider decentralized aggregation protocols to establish communication to these nodes, to continuously update that data, and to decide when to offload data. The gossip-based approaches that have been successfully applied to peer-to-peer systems [3] are adapted to work in opportunistic and distributed manner [4].

4.3 Heterogeneous Opportunistic Communications

Our objective is to develop a radically different communication paradigm than existing networking approaches have: dedicated endpoints (source/sink) do not exist, data is not forwarded, but aggregated and an omnipresent distributed knowledge base is created. This consists of mainly 2 components of (1) the neighbourhood management and (2) the selective transmission of information to neighbours (e.g., it has to be decided, which information is to be sent at which time to which neighbors). For the **neighborhood management** the following requirements are being considered:

- The communication technology is heterogeneous: e.g. WiFi Direct, Bluetooth and NFC for smartphones and IEEE 802.15.4 and Bluetooth for WSNs.
- Most devices are battery powered, therefore energy efficiency is an important requirement.
- Support of cecurity, privacy and anonymity schemes of the higher layers.

The second component of **selective transmission of information** is designed based on some ideas on caching and forwarding strategies from Information Centric Networking (ICN) [5] based architectures. Once the neighbors are discovered, one-hop data communications between neighbors are initiated. We focus on the content that parties need to exchange instead of the hosts on which the content resides. The content itself is named and this is ideally suited for the DICE neighborhood as the participants' focus in this environment is simply the

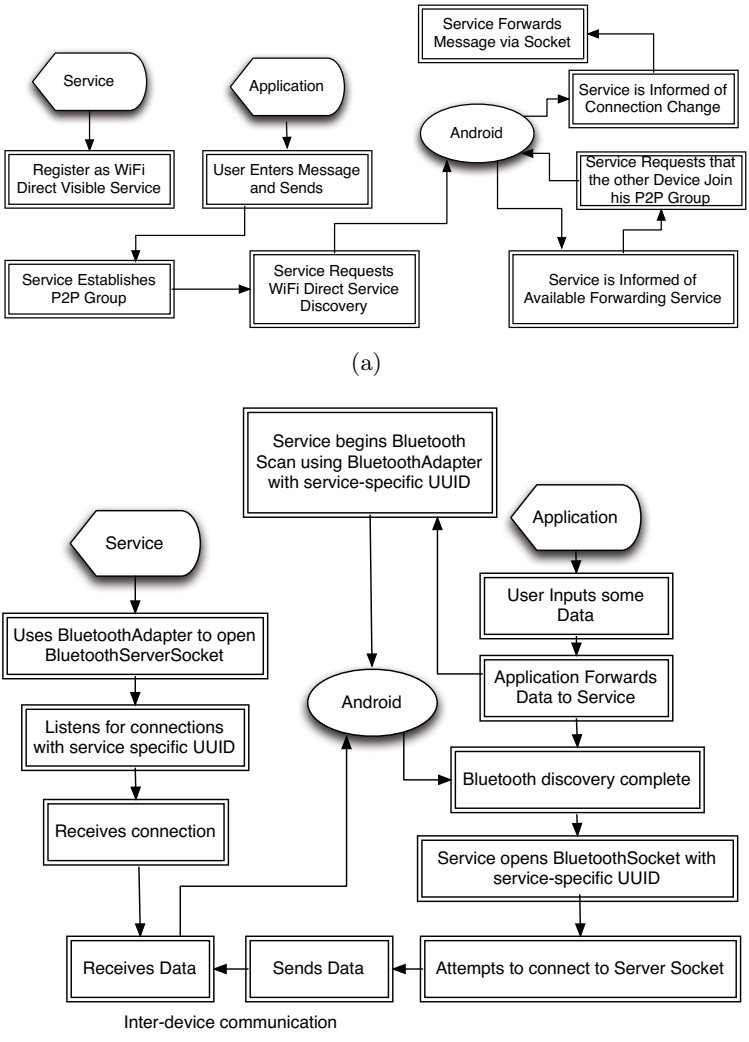

Fig. 3. First Prototype of Heterogeneous Opportunistic Communication Module using WiFi Direct and Bluetooth

information itself. The information is disseminated in a distributed environment to support the selection of suitable neighbors based on the amount or quality of available data in the nodes (instead of the distance to the sink) [6].

Figure 3(a) and 3(b) show our first prototype implementation of **Heterogeneous Opportunistic Communication** module on the Android platform. The WiFi Direct and Bluetooth communication technologies are used in the initial implementation. We use extremely simple forwarding protocol. When a device wants to send a message, or receives a message it hasn't seen before, it

will simply forward the message to all of its peers who have also installed the DICE service. The WiFi Direct operates in infrastructure mode, which affords it a number of advantages over ad-hoc mode.

As shown in Figure 3(a), the Android **WifiP2pManager** class provides a number of relatively simple functions for identifying and connecting with peers. Once a connection has been established, **WifiP2pInfo** object provides the necessary information to open a socket with the group owner. Although WiFi Direct requires Android API level 14, legacy devices can still connect to an existing WiFi Direct network as if it were a regular access point. This may allow us to extend the app to support older devices in the future. The BluetoothAdapter is the primary object used in Android's Bluetooth API. It represents the device's physical Bluetooth Adapter, and enables device discovery, changing bluetooth visibility, and accepting connections. As shown in figure 3(b), upon the completion of Bluetooth discovery, Android provides us with a list of BluetoothDevices, which allows us to open a BluetoothSocket to connect with a remote device.

The current implementation will be extended to support the following features of **Heterogeneous Opportunistic Communication** of DICE.

- Develop a reliable neighbor discovery protocol: neighboring participants in a DICE environment across heterogeneous network technologies (WiFi Direct, Bluetooth, NFC) need to be identified, updates of the neighborhood need to be detected with minimum signalling. Therefore different modes need to be supported, e.g., push/pull. This means DICE nodes can announce their presence as well as they can probe their environment to retrieve information.
- Develop a smart broadcasting scheme making use of the wireless broadcast advantage for information distribution.
- Develop algorithms to predict the quality of a link to a neighbour in means of link stability and link quality, but also in relation to the amount and quality of available data.
- Develop methods (gateway concepts) to communicate between neighbors with heterogeneous link layer technologies, e.g. connect sensor nodes with IEEE 802.15.4 to smartphones with WiFi access.
- Develop an efficient data link-layer protocol: the signalling overhead needs to be kept to a minimum to optimize performance and energy-efficiency, still the reliability of the transmission between neighbors needs to be maintained.
- Develop schemes supporting the information security realized in the the higher layers: e.g., encryption on link layer, link-layer protection against modification
- Support anonymity of data already in link-layer: this needs to be considered together with higher layers functions, but it needs to be ensured that link layer protocols do not contradict higher layer anonymity of data.
- Develop a smart forwarding protocol: The features such as named based information dissemination, caching at different points and transmitting of the information, specially in a distributed environment should be considered.

4.4 Data Security and Safety

The DICE collects and propagates data from several users. This data consists of the primary data on the one hand, but also of derived data. The protection of this information must be assured by trustful handling without being too restrictive, as the usage of this (derived) information is the key to having handy and useful applications. However, the innovative concept of anonymous, infrastructureless data transfer and storage in DICE leads to new security threats:

- How to ensure no users are acting purely selfishly and only consuming data from the DICE instead of also contributing and forwarding?
- How to prevent an application or any third party from manipulating the data used by other applications in order to distort the data base? As there is no global server the data that is sent by devices nearby cannot be validated and must therefore be trusted.
- How to guarantee anonymity on the one hand and prevent abuse of the system on the other hand?

Though we do not focus on implementing very sophisticated security features for the first prototype, we will concentrate on lightweight (in term of memory and CPU usage) protocols and energy efficient implementations. Particular attention will be given to design interoperability between devices with very different capabilities, thus the need to support negotiations between different cryptographic and security parameters. Our first prototype implementation will be open to possible trade-offs between degree of security and system efficiency.

5 Application Ecosystem

In the previous section we have presented the implementation design of our DICE architecture. However, its usage and the possible business models behind it are as important as its functionality or performance. In summary, DICE is and will be implemented as an open-source community effort, so that everybody can take advantage of it everywhere. Any other business model for the main supporting architecture is unthinkable because of the targeted applications. However, DICE provides only the platform, not the data and not the user-oriented applications. At the same time, we assume that data emerges from usage: i.e. if you want to use a DICE service, you need to also produce/sense data. Thus, the remaining question is which kind of applications DICE is able to support and what are their underlying business models. We differentiate between:

End-user simple applications. These applications are end-user oriented and visualise or generally consume directly the data provided by other users. For example, a map representation of the noise levels around the current location of a user or a list representation of the closest vegetarian restaurants. The shared properties are simple visualization/representation in a map/list/table and simple processing of the available data by summarising, averaging, etc.

These applications are simple to implement. However, different users have different tastes and preferences for how a particular piece of data should be represented. For example, some people prefer generally using a map, some prefer using a list. Additionally, while the processing of the data is trivial, there are also billions of possibilities how to combine it. For example, Alice might want to see all vegetarian or biological restaurants around but also all Jazz bars, but Bob might be interested in all non-Asian grill restaurants.

Thus, the solution is to provide the users directly with the possibility to "implement" their data consumption applications in a building-block style. There, several visualisation options are provided together with a free selection of data filters. Not only that the user will be able to customise their own applications, they will also be able to exchange the applications themselves through the DICE platform. This will further motivate end users to participate in the process and increase the attractiveness of the platform, keeping it at zero costs for the users. The building-block style application programming will be part of the open-source community effort, to make sure again that the service can reach all citizens.

Sophisticated or specialised applications. Some applications will require a more sophisticated data mining processing behind the scenes than simple data presentation. For example, all reports arriving from citizens about their environment, such as broken street lamps, need to carefully recognised to make it more comfortable to the end users. For example, a typical report would be "broken street lamp!" or "flowers need some more water here!". These reports will have a geospatial timestamp to localise the problem, but no tags nor meta data. Thus, the application running on the smartphone of the gardening brigade in the city needs to filter all messages relevant to it, irrespective of the language or the exact wording they use. This will require a sophisticated data mining approach.

However, also other usages are possible, such as using DICE as the main traditional-style communication platform in case of emergencies of disasters. In this case, a specialised application is needed to serve the needs of fire brigades, police or volunteers. Thus, such applications become the task of professional programmers and data mining experts, who can sell their products to local authorities, industries or organisations, who need more complex data processing or non-standard solutions.

Advertisements. Another possible usage of the DICE platform is to launch advertisements or other paid information to local users. For example, a small shop in the city centre might decide to send a 20% coupon to all users around to draw their attention and to come in. The payment option makes it possible to keep the advertisement for a longer time near the shop or to give it priority when propagated within the DICE platform.

The above examples show in how many different ways such a platform can be used and how many novel business models can arise form it especially for small, medium and startup businesses. Similarly to the app boom in all currently available app stores, DICE is able to generate a whole new ecosystem of applications and services at extremely low cost for end users.

6 Conclusion and Outlook

In this paper, we have discussed a feasible implementation design for a new approach to cloud computing, called the Massively Distributed Cloud (MDC). The DICE architecture is completely independent from communication infrastructures and instead uses direct communications between devices to exchange relevant data. Services in this cloud are also distributed, where users are able to define their very own requirements. The DICE architecture presented here is under active development for the Android platform.

As future work, both simulative and theoretical analysis will be done to evaluate DICE platform and its services. The objective is to do a requirement analysis (in terms of requirements in communication and networking, resource management, security, data management, etc) to investigate how the proposed architecture can be adapted and scalable in different application scenarios. Further, our implementation work should be extended for heterogenous platforms (Android, IoS, WSN, etc) and devices (smart phones, tablets, sensors, etc). Last but not least, we will launch a real user pilot study to evaluate the social attractiveness of this novel paradigm and better meet user requirements and expectations.

Acknowledgements. We would like to thank our European colleagues and friends, Amy L. Murphy from FBK Trento, Italy; Alberto Montresor from the University of Trento, Italy; Axel Wegner from TuTech Innovation GmbH; Frank Bittner, Ulfert Nehmiz and Mehmet Kus from Otaris Interactive Services, Germany; Mischa Dohler from WorldSensing Inc., Spain and Oliver Gerstheimer from chilli mind GmbH, Germany. The presented vision in this paper has been the result of fruitful and interesting discussions with these people. Further, we thank Isaac Supeene for the implementation work done in communication module during his stay at the University of Bremen as an exchange student.

References

1. Förster, A., Kuladinithi, K., Timm-Giel, A., Görg, C., Giordano, S.: Towards the massively distributed cloud. Under submission (2013)
2. Garg, K., Giordano, S., Förster, A.: A study to understand the impact of node density on data dissemination time in opportunistic networks (in under review, 2013)
3. Jelasity, M., Montresor, A., Babaoglu, O.: Gossip-based aggregation in large dynamic networks. ACM Trans. Comput. Syst. 23(1), 219–252 (2005)
4. Guerrieri, A., Carreras, I., De Pellegrini, F., Montresor, A., Miorandi, D.: Distributed estimation of global parameters in delay-tolerant networks. Computer Communications 33(13) (August 2010)
5. Jacobson, V., Smetters, D.K., Thornton, J.D., Plass, M.F., Briggs, N.H., Braynard, R.L.: Networking named content. In: Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies, CoNEXT 2009, pp. 1–12 (2009)
6. Wenning, B., Lukosius, A., TimmGiel, A., Görg, C., Tomic, S.: Opportunistic distance-aware routing in multi-sink mobile wireless sensor networks. In: Proceedings of the ICT-MobileSummit (2008)