Architecture for Platform- and Hardware-independent Mesh Networks

How to unify the channels

Sebastian Damm, Michael Rahier, Thomas Ritz, Thomas Schäfer
Mobile Media and Communication Lab, FH Aachen
Aachen, Germany
s.damm@fh-aachen.de
rahier@fh-aachen.de
ritz@fh-aachen.de
thomas.schaefer@alumni.fh-aachen.de

***Abstract*—This paper will prove that mesh networks among different platforms and hardware channels can help to channel valuable information even if public telecommunication infrastructure is not available due to arbitrary reasons. Therefore, results of a simulation for mesh networks on mass events will be provided, followed by the developed architecture and an outlook on future research. The developed architecture is currently being implemented and field tested on mass events.**

Keywords-mesh networks; platform independence; mobile software.

I. Introduction

On mass events like music festivals, the cellular reception is often insufficient because Global System for Mobile Communications (GSM) cells tend to be overloaded. Especially in terms of security, this is a serious issue, as people might not be able to communicate with rescue forces. This can lead to catastrophes like the mass panic on the German Love Parade in 2010 where 21 people died. During investigation of this event, crowd scientist G. K. Still pointed out that missing communication was one of the key factors for what happened [1].

Nowadays, we are dealing with plenty of mobile hard- and software platforms like iOS, Android, Windows Phone and others. These platforms use all different kinds of connection channels such as Bluetooth, WiFi(-Direct), NFC, etc. That means that there are several possibilities to compensate the mentioned lack of connectivity. Mesh networks can be a solution where people keep connected on such mass events, without having any connection to a cellular network. With support of some well-placed infrastructure like access points, relevant data could be pushed into the crowd and then be routed or broadcasted to other persons from device to device (Fig. 1).

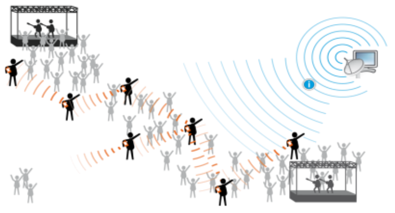

Figure 1: Illustration of information flow in the crowd

Unfortunately, there is no or just few integration of the different communication channels, even within single platforms, and there are even more difficulties when trying to interconnect different platforms. To negotiate these obstacles, this paper presents a platform and hardware-independent architecture that integrates all different network types into an abstract layer. This architecture is currently being implemented as a java base library, which is used on Android. There is also an implementation for Windows Phone. Further research on the possibility of an iOS implementation is ongoing. The main goal is to enable automatic interconnectivity between different mobile platforms without the need of user interaction and regardless to the used communication technology. This will provide better ways for promoters of mass events to reach their guests in case of emergency.

Following to this introduction, the paper will show related work in the field of mesh networks. In section three, a simulation will be presented, showing the possibility of creating a mesh network in the scenario of a mass event. After the general possibility is proven, an architecture for hardware- and software-independent mesh networks will be introduced in section four, followed by a conclusion and outlook to future research.

II. Related Work

The research project "iWave" (information waves on mass events) addresses the problem of missing or poor connectivity on mass events. No connection means no ability to communicate in security related issues like mass panic, severe weather or just injuries. The architecture proposed in this article is part of ongoing research, where a reusable communicator component to span mesh networks will be implemented on different mobile platforms.

One project providing similar functionality is the middleware "Beddernet" [2]. It is capable of spanning mesh networks using Bluetooth. Unfortunately, there are several downsides of Beddernet. First, it is limited to Android and not available for other platforms. Second, it is using Bluetooth as the only channel, leaving out WiFi and such. The third problem is that the last change to the project was committed in August 2012 (and before that in July 2010); so, it can be assumed the project is not maintained anymore.

Another project dealing with the mentioned issues is the MANET project [3]. It uses sensor nodes based on the ZigBee standard (based on IEEE 802.15.4) [4]. A huge disadvantage is the use of special sensor nodes instead of the built in hardware in smartphones that people already carry around.

The University of Darmstadt proposed an approach to use WiFi routers in emergency cases to span ad-hoc networks [5]. This is suitable for communication within cities, but not transferrable to mass events, as the area is quite limited but crowded with lots of people with not as many routers as in an urban environment.

All these projects just address one communication channel; they are not updated anymore and/or need specific hardware. Currently, there is no project that abstracts from the hardware and unifies all different kinds of channels provided by modern smartphone hardware to create a communication layer that is transparent to the user. This fact raises the question, if it is possible to form a mesh network in such environments in general.

III. Simulation of Mesh networks

To answer the question raised in the previous paragraph, the following simulation was implemented.

A. *Simulation set-up*

To evaluate if spanning a mesh network in an environment like a music festival is possible, a simulation of the scenario was conducted using the following steps:

1. Creating a model of the area
2. Generating and distributing nodes
3. Generating edges
4. Analysis

In the first step, a model of the whole area is created. It is divided into sub-areas with varying priorities for the density of people. For example, the area in front of a stage tends to be more crowded than a tent with merchandising. An example for an area with different priorities can be found in Figure 2. The upper left corner will have two times as much people in it and the lower right six times more than the rest of the area.

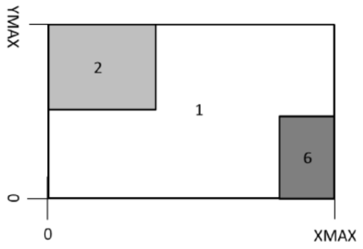

Figure 2: Example for an area model

Second step is to generate nodes which is equivalent to an amount of people. To achieve uniform distribution of the nodes in each subarea, X- and Y- coordinates of a node are represented by random numbers between zero and Xmax/Ymax.

Now, the priorities of each area are used as the probabilities for a Bernouilli experiment [6]. The result of this experiment determines whether or not a randomly generated node is added to the subarea.

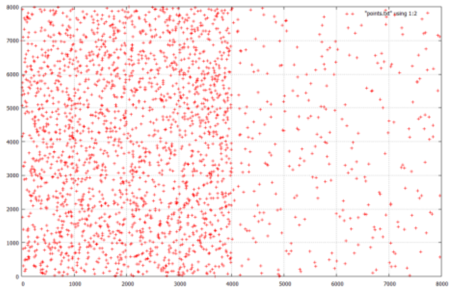

Figure 3: Example plot of distributed nodes

In Fig. 3, the distribution of nodes in an area where the priority on the left is four times as big as in the right half is shown.

After that, the generation of edges between the nodes takes place in the third step. Target of this step is to find out, how many mesh networks could possibly exist within the whole area. To create a mesh network, specific parameters of a communication interface need to be taken into consideration, such as maximum range and maximum number of connections per interface. The range can either be set to a fixed value like the maximum range in the device specification or vary within a certain codomain. The first will create optimistic and the latter pessimistic results. It turns out that the pessimistic results are more realistic, as first tests for Bluetooth pointed out, that with only few people the maximum range of 100m can be reached, but within a huge crowd with lots of devices interfering, it can only be a few meters. Using these values, the approach is as follows:

Starting from a random node, it first will be checked, if this node has reached its maximum connection count. When there are still connections available, the node will be compared to its neighbors. If a neighbor is within the interfaces range and also has connections available, an edge between the two nodes will be created. This will be repeated until there is either no node with free connections left, or all nodes are processed. Output of the third step is an undirected graph.

The final step is to analyze the resulting graph. Using repeated breadth first search until all nodes are marked, it can be determined of how many connected components the graph consists of. This represents the number of possible mesh networks in the modelled area depending on the number of people and specified device parameters.

To receive meaningful results, 22 iterations from 400 to 1500 nodes with increments of 50 were realized. Due to the fact that nodes are placed randomly within the areas, each iteration was executed 10000 times to get good average values. We considered a connection count of seven (active) connections for Bluetooth and an optimistic range of 50m (half of the specified maximum). The results for a model area of the German music festival “Das Fest” [7] will follow in the next section.

B. Results and Discussion

The probability of connecting all nodes to one single mesh network raises with the number of nodes. Above 1000 nodes, the chance is higher than 90% and gets close to 100% for more than 1300 nodes. Even with only 400 nodes, there are not much more than two separated networks and just around one node without any connection. These numbers lower drastically with more nodes added to the area. A reason, why a full connection of all nodes is not possible is, that the maximum number of connections for each node will be reached at some point.

The values of the optimistic simulation changed drastically if the range is changed to dynamic values between 5m to 100m, depending on the density of people/devices in an area:

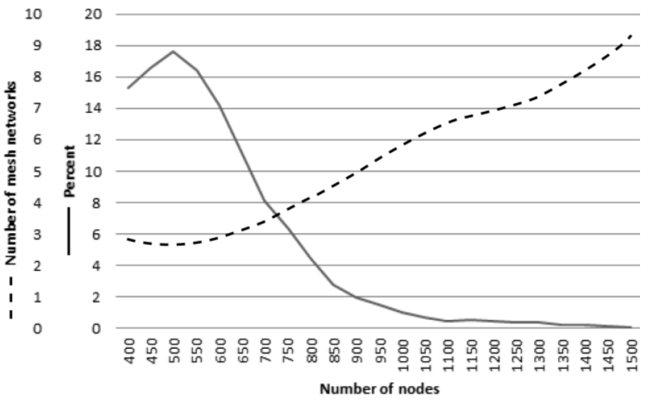

Figure 4: Probability of a single mesh network / Number of networks

As Fig. 4 shows, after a raising (but in comparison still low) probability, it decreases significant for more than 500 nodes and almost reaches 0 for 1200. The number of mesh networks will raise up to 9.5 for a number of 1500 nodes.

The results of both simulations show, that it is possible to create mesh networks within mass events. Even though it is likely to get more than one network, it should be possible to compensate this using only little, well placed infrastructure to connect the different subnets.

This first simulation focused on Bluetooth, but it can be easily adapted to WiFi(-Direct) by changing number of connections and range. Further results are expected within a short timeframe.

IV. A PLATFORM- AND HARDWARE-INDEPENDENT ARCHITECTURE

As shown in related work, there is no real treat to the current issues, but as seen above, the possibility to reach many people on a mass event using mesh networks is given. Therefore, a new architecture for the implementation of mobile mesh networks will be introduced. After pointing out the requirements for such architecture, the different layers will be explained following by an overview of the complete architecture.

A. Claims to the architecture

Goal for the architecture is to abstract hardware and software-platform of mobile devices and enable automatic connection and routing between these. All higher layers should just know, there is a way to communicate, regardless, which specific one it is.

Each component of the architecture should be encapsulated and separated strictly from other components so that the architecture keeps being flexible and customizable without high efforts. The routing e.g. should not correlate with any hardware specific implementations so that a new routing algorithm can be integrated, without touching other code but the router itself.

Finally, the whole architecture and its implementations should be easy to integrate into mobile apps by providing a well-defined interface with just few methods and events.

B. Architecture Layers

The architecture consists of four layers that are independent from each other and only communicate via messages/events and it provides an interface which encapsulates all layers. Thus, each layer can be implemented separately and exchanged by different implementations.

The bottom layer is the datalink layer. It contains all hardware specific code and manages the connections for the different channels. The connector components for each datalink automatically search for available peers and try to connect to them. Once a connection is established, the IO-Stream will be passed to the next layer - the Local Peer Manager.

This layer holds all used datalinks and names for the peers. From here on, the system only deals with names and does not care about which hardware channel is used anymore. Instead it just receives the given streams and forwards them to the Message Broker.

The Message Broker is responsible for parsing incoming byte streams and distinguishing between routing messages and text messages. Routing messages will be deserialized and handed to the routing layer to control the message flow. Text messages will be handed to the router without being touched.

Currently Ad-hoc On-Demand Distance Vector (AODV) [8] routing is used; but, due to the independence of the layers, it could easily be exchanged with Destination-Sequenced Distance Vector (DSDV) [9] or any other routing protocol.

The iWave Communicator surrounds the layers of the whole architecture as a façade. It provides simple functionality to control and reuse the architecture, such as events for new connections and disconnections, listing all available peers as well as methods to send messages or broadcast them to the whole network.

After introducing all layers and components, the following Fig. 5 will provide an overview over the complete architecture and coherences.

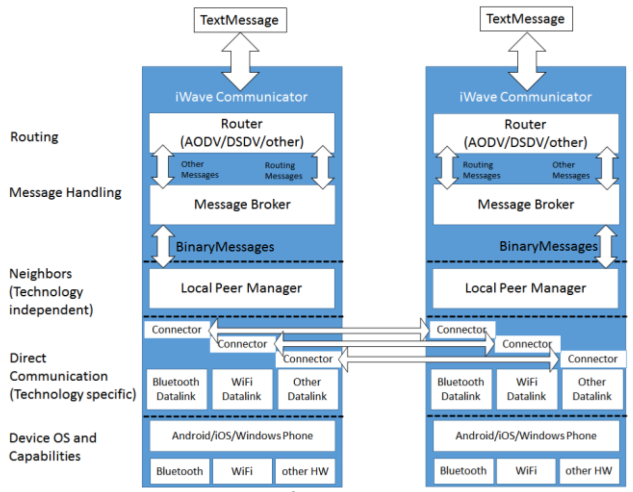

Figure 5: Architecture overview

The given architecture is currently under evaluation and first implementations will be tested soon.

V. CONCLUSIONS

After introducing the need for a hard- and software independent architecture and taking a look at existing approaches it was proven in a simulation, that it is possible to create mesh networks in environments of mass events. Finally, this paper introduced a hard- and software independent architecture to be used on all mobile platforms for this purpose.

First implementations of this architecture on Android and Windows Phone have shown that current mobile platforms are all more or less restricted when it comes to interconnectivity. This makes it hard to achieve the goal of the architecture being totally independent of the platforms.

Android seems to have the fewest restrictions right now. It is able to initiate outgoing as well as accepting incoming connections. Using "InsecureBluetooth" it is even possible to connect two devices without the need of manually pairing the devices, if they are running the same app. For WiFi-Direct there are workarounds to avoid the need for user interaction via hidden API calls and there is also an ongoing discussion of providing an official API call to do so [10].

Windows Phone 7 does not allow to control Bluetooth programmatically at all and WiFi-Direct connections are just allowed outgoing. Windows Phone 8 does not allow any incoming Bluetooth/WiFi connections from non-Windows devices. However, the Phone 8 SDK contains a class called PeerFinder [11] to automate search and connection between two Windows Phone devices at least via Bluetooth. Even though it also contains a property "AllowWifiDirect", this is not supported so far [12].

The possibilities for iOS are still under evaluation and while writing this paper, Apple announced iOS7 [13] which brings a new framework called "Multipeer Connectivity Framework". It is supposed to enable ad-hoc connections between iOS devices. Unfortunately, it seems like this will not work between iOS and other platforms.

VI. OUTLOOK

Due to the mentioned restrictions, further research has to be done to work around these issues and provide connectivity between all different platforms. The vendors should open up their platforms to developers a little more because especially for security related messaging, meshed networks could be ideal. There is clearly the need for a common API and a standardized message format to enable this seamless connectivity across platforms.

After promising results in a lab environment, the first Bluetooth implementation of the proposed architecture was field tested in July 2013 at the music festival "Das Fest" in Karlsruhe, Germany [7] and the collected data is currently being analyzed.

ACKNOWLEDGMENT

Thanks to the Federal Ministry of Education and Research for supporting the project "iWave" (program: "Central Innovation Program SME", ref. no. KF2457707ED2).

REFERENCES

[1] G. K. Still, "Duisburg - 24th July 2010 Love Parade Incident - Expert Report," FIMA Bucks New University, Dec. 2011.

[2] R. Gohs, S. Gunnarsson, and A. Glenstrup, "Beddernet: Application-Level Platform-Agnostic MANETs", in LNCS, Distributed Applications and Interoperable Systems, P. Felber and R. Rouvoy, Eds.: Springer, 2011, pp. 165-178.

[3] Forschungszentrum Informatik, MANET Projekt. Available: http://www.manet-projekt.de Retreived: 10.08.2013.

[4] S. Farahani, ZigBee Wireless Networks and Transceivers. Newton, MA, USA: Newnes, 2008.

[5] K. Panitzek et al., "Can We Use Your Router, Please?: Benefits and Implications of an Emergency Switch for Wireless Routers," Int. Journal of Information Systems for Crisis Response and Management, vol. 4, 2012, pp. 59–70.

[6] J. V. Uspensky, Introduction to mathematical probability. New York etc: McGraw-Hill, 1937.

[7] Das Fest GmbH, Das Fest - Official Website. Available: http://www.dasfest.net. Retreived 08, 2013.

[8] C. Perkins, E. Belding-Royer, and S. Das, Ad hoc On-Demand Distance Vector (AODV) Routing. US: RFC Editor.

[9] C. E. Perkins and P. Bhagwat, "Highly dynamic Destination-Sequenced Distance-Vector routing (DSDV) for mobile computers," SIGCOMM Comput. Commun. Rev, vol. 24, no. 4, 1994, pp. 234–244.

[10] Google, Issue 30880: Wi-Fi Direct API for connection acceptance. Available: https://code.google.com/p/android/issues/detail?id=30880. Retreived: 10.08.2013.

[11] Microsoft, PeerFinder Class. Available: http://msdn.microsoft.com/en-us/library/windows/apps/windows.networking.proximity.peerfinder. Retreived: 10.08.2013.

[12] Microsoft, PeerFinder.AllowWiFiDirect. Available: http://msdn.microsoft.com/en-us/library/windows/apps/windows.networking.proximity.peerfinder.allowwifidirect. Retreived:10.08.2013.

[13] Apple Inc, iOS 7 beta for Developers. Available: https://developer.apple.com/ios7/. Retreived: 10.08.2013.