# Platform Migration and Performance Enhancements of the Lifeline Prototype

Semester Thesis ETH Zürich

Andreas Wapf
awapf@student.ethz.ch

September 19, 2007

# Contents

# Acknowledgements

Many thanks to all the people who made it possible for me to write this thesis here at the National Institute of Standards and Technology (NIST) in Gaithersburg, MD, USA. Without the support of all this people it would never have been possible for me to have such a great experience and to learn so many new things in such a short amount of time.

Special thanks to Prof. Roger Wattenhofer at ETH Zürich for his flexibility. Without his announcement of the internship offer on the ETH mailing list and support during my time in the USA I would not be here and write my thesis in such a great country.

Many thanks go to my super visor Dr. Nader Moayeri who gave me the opportunity writing my thesis at NIST.

Special thanks also to Dr. Michael Souryal who always encouraged me. Without his tipps and interesting conversations this work would not be the same.

Last but not least I would like to thank all other persons who made my journey here so interesting.

# 1 Introduction

## 1.1 Motivation

Current first responder radio communication technologies are susceptible to interruption and disconnection due to propagation loss of the radio signal with distance and obstructions, especially in large buildings with metal construction. Multihop networks use intermediate nodes to relay the signal over multiple, shorter hops from source to destination. By constantly measuring the state of the wireless link, inexpensive relays can be dropped as "breadcrumbs" at appropriate points along a first responder's path into a building to maintain multihop communications with the outside. Challenges include developing accurate real-time link state measurement techniques, determining the optimal placement of wireless breadcrumbs, and adapting to time-varying environmental conditions to maintain end-to-end connectivity. The breadcrumb system is not just restricted to first responder scenarios and could also be used as an example for miners, robots etc.

Hannes Geissbühler of ETH Zürich built a prototype of such a reliable ad hoc multihop communication system based on the low-speed 900MHz Mica2 Crossbow Motes in connection with his masters thesis [8]. Currently, available services are text messaging and the transmission of sensor values. For a first responder a text interface is inconvenient because he is not able to write messages during a mission. A more practical form of communication with incident command is voice.

Crossbow's Mica2 Motes are very limited because they are optimized for operation over long periods of time such as in wireless sensor networks. They are limited to a maximum bandwidth of 38.4Kbits/s and the round trip time in larger networks increases fast due to the limited processing speed of the processor on the nodes. In order to provide a real time voice communication capability, it seems inevitable finding an alternative platform.

A higher bandwidth and a shorter end-to-end delay would not just allow transferring audio data, but they also make it possible to send pictures and even video streams much faster to the other end of the communication line.

Another problem with the current prototype is that it supports just one first responder. However, a typical rescue mission consists of a group of persons with communications needs.

In summary, the goals of this thesis consist of evaluating a suitable new platform, porting the already available services to the new platform and adding the functionality of audio communication and the support for multiple first responders.

## 1.2 Overview

In the following chapters the entire system building process is described. Section 2 describes the type of hardware and software platform selected and what the main drivers for these choices were. Afterwards section 3 is about the deployment algorithm and the routing protocol. The deployment algorithm allows to build the

network in real time while the routing protocol organizes the end to end communication. A evaluation of the network is following which includes the measurement of the throughput and delay from the base station to a mobile user system. Finally the last section deals with the migration of the existing applications and the introduction of audio streams into the LifeLine system.

# 2 Platform Selection

The first step was to decide about the hardware. The priority was finding an appropriate wireless technology to have a bandwidth high enough to support at least audio transmissions. A high bandwidth makes it possible to have multiple phone calls at the same time or even video transmissions.

Based on the chosen wireless platform, the next step was to determine the controller for the wireless module. In other words it was necessary to choose a motherboard with enough CPU power and memory to operate the WiFi module and relay all the traffic. The decisions were mainly driven by the availability, bandwidth and ease of integration.

## 2.1 Hardware

### 2.1.1 Wireless Technology Selection

For the feasibility of the audio streaming it is inevitable to have enough bandwidth. But the used bandwidth was hardly unpredictable and depends on the routing protocol traffic, the number of applications and last but not least the number of mobile user systems in the network. In general as more bandwidth is available as more applications and user can run in parallel. A second factor besides the bandwidth is the availability and the small form factor. The final product should not be too heavy or too expensive.

The number of possibilities decreased dramatically after this requirement definition. The only readily available system with a bandwidth high enough for this application appeared to be the 802.11b/g technology. Technologies like ZigBee[1], Bluetooth[2] or a Nordic Chip[3] have all some disadventages. These systems are either to slow, need to much work to integrate them or have a too short range. Although Hannes Geissbühler mentioned possible problems with the chosen architecture in his thesis[8] - especially in the ad hoc mode - the expected challenges seemed to be solvable.

The first trials with 802.11b/g cards were deflating. The instability of the driver and also the hardware made more problems then expected, especially in the not so well tested ad hoc mode (described later). Several wireless cards work fine with the right hardware, firmware and driver combination only. It is hard to debug the drivers while the errors show up sporadically. It took a long time to figure out the right combinations. Some of the cards just do not work or run just fine until they reach a certain amount of traffic.

After testing a lot of different cards, the following list shows the most stable combinations:

[1] http://www.zigbee.org
[2] http://www.bluetooth.com
[3] http://www.nordicsemi.no

| | | | |
|---|---|---|---|
| WiFi Card: | Netgear MA701 | SMC 2642W | wifistix |
| HW Version: | 1.0 | 2.1 | (details later) |
| Firmware Version: | 1.4.9 | 1.3.6 | |
| Linux Kenel: | 2.6.17 | 2.6.17 | 2.6.17 |
| Driver: | HostAP 0.4 | HostAP 0.4.4 | Marvell |

### 2.1.2 Overview

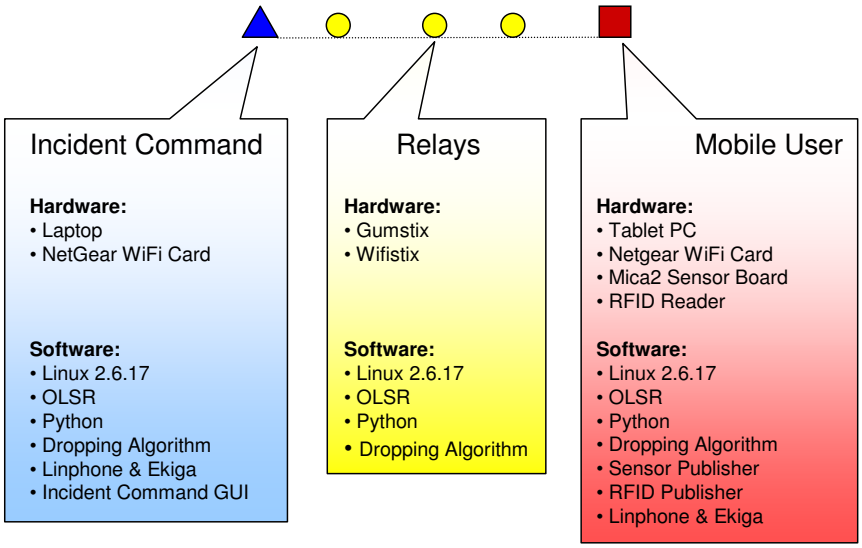



Figure 1: Hardware and Software Overview

As in the previous work the project consists of three different kinds of systems. The mobile user (i.e. a first responder, a robot or similar) that goes into a area with a bad propagation behavior and carries the "Mobile User System" and several "Relay Systems". The mobile user system acts as an interface to the "Base Station System" over the network build out of dropped Relay Systems.

### 2.1.3 Mobile User System

The main hardware components for the mobile user are the CPU with the wireless module, the radio-frequency identification (RFID) reader for the localisation part and the vital sign sensors.

In the old system a Compaq IPAQ H3800 with a running Java Virtual Machine was the solution to connect everything together. Bad experiences with the stability of the IPAQ computers during presentations forced to use another device. Another reason to change the system was the inappropriate touchscreen keyboard on the

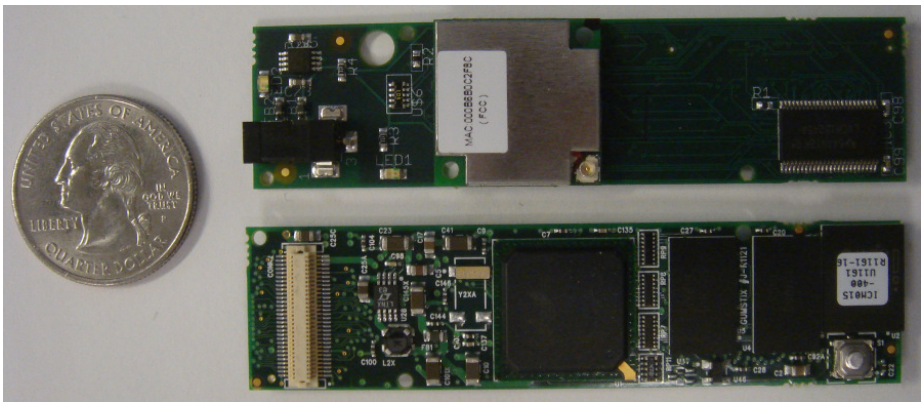


Figure 2: Gumstix (bottom) and Wifistix (top)

IPAQ. An Intel x86 compatible tablet PC with a PCMCIA and Compact Flash card slot for the wireless card and the RFID reader is the replacement. It is *not* supposed to be the best solution for the mobile user. The emphasis in this work is on communication part and the proof of concepts, not an end-user integration.

The RFID reader is the same as used in the first prototype, the ACG Dual ISO CF Card Reader, for details see [9].

The sensor values come from the same MTS310 sensor board as in the old system and it is connected to the tablet PC over a mica2 node and USB connection. Further details can be found in the Crossbow data sheet under [7].

### 2.1.4 Relay System

The main requirements are a computer, a WiFi module, mobility and size. The system should be as small as possible because the fire fighter does not need any extra unnecessary burdens.

A problem with WiFi cards is that they are not optimized for a use in wireless sensor networks. It was hard to find tiny mobile wireless LAN nodes. For the first implementation a couple of Compaq iPAQs 3800 with a special expansion pack for PCMCIA cards were used. After a deeper search for a smaller platform the gumstix computer system [2] from the company of the same name turned out to be the right choice.

The gumstix computers as seen in figure 2 have a tiny form factor and they can be expanded by different daughter boards. The **connex 400xm**[1] series consist of Marvell's (formerly Intel's) PXA255 driven motherboards and were used in this project.

| | |
|---|---|
| Name: | Gumstix connex 400xm |
| Processor: | 400Mhz Intel XScale®PXA255 |
| Memory: | 16M Flash, 64M SDRAM |
| Size: | 80mm x 20mm |

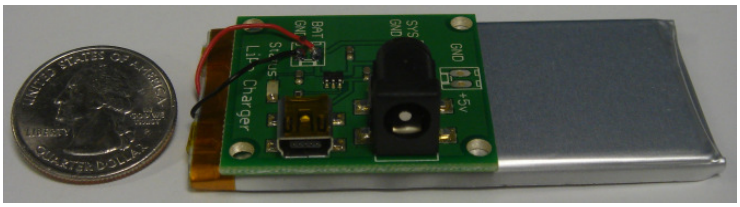



Figure 3: Lithium Polymer Battery with Charger Board

There is also a compatible WiFi daughter board called **wifistix**[4] with the following specifications in the same form factor available:

| | |
|---|---|
| Name: | Wifistix FCC |
| Features: | 802.11(b) and 802.11(g) |
| Chip: | Marvell®88W8385 |
| Channels: | 1-11 |
| Connector: | External Hirose U.FL ultra-miniature coaxial receptacle |
| Size: | 80mm x 20mm |

The prototype requires that the relays are mobile. The manufacture of the gumstix boards delivers just wall adapters, so a rechargeable lithium polymer **battery** with a LiPo charger[5] solved the problem. The LiPo charger makes it possible to charge the battery in place and connect and disconnect the entire system from the wall adapter without a restart of the node.

LiPo batteries are usually dangerous in terms of under or over voltages. The battery has a under voltage protection and the charger board an over voltage protection. As long as the battery is in the right temperature range nothing dangerous should happen.

Lithium polymer batteries are simple to use and have no memory effects. For special environments it is necessary to adapt this solution. It is important to find another less temperature dependent batteries for high temperature environments. For detailed information about batteries the battery university is website[4] helpful.

The system is powered by a 930mAh UltraLife Lithium Polymer **battery**[6] which can power the gumstix in combination with the wifistix for around 2 hours.

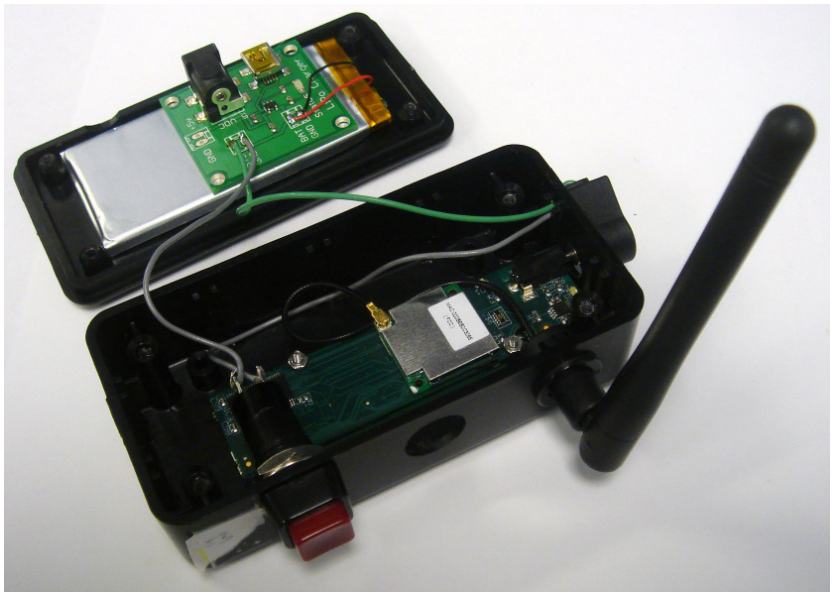
Everything together fits into a small plastic box of 1x2x4 inches with an external antenna and an on and off button.

### 2.1.5 Base Station

In terms of hardware the only change in comparison to the first prototype is the additional wireless card and the abandonment of the USB/MIB520 connected Mica2 note. The same standard Dell laptop is used as before.

[4] http://www.batteryuniversity.com/

(a) Outside of the Box



(b) Inside of the box

Figure 4: Mobile User System with External WiFi Antenna

## 2.2 Software Platform

The old system was based on TinyOS on the Mica2 motes and Windows 2000 or Mobile 2003 at the network edges. During prototyping it is very helpful to have good debugging facilities. It was not a problem on the incident command or first responder site but awkward on the intermediate Mica2 motes. They had just 3 different LEDs for debugging purposes. Another stumbling block was a missing protection mode and the inability to allocate dynamically memory on the motes. As soon as there is an execution error the entire node does not work anymore. The lack of a standardized network stack required to implement all kind of tools like *ping(8)* or message tools.

These problems were gone after the decision to use Linux on all systems. The gumstix computers are supported by the Linux 2.6 kernel[5]. All the software parts are integrated based on the Linux Kernel version 2.6.17. Linux has a standard IP stack with TCP or UDP. Standard tools like *ping(8)*, *tracepath(8)* and *iperf(1)* among others are available out of the box.

Almost all code is implemented in the python programming language[6]. Although it would be expected for such a system with low level access to sockets and network stacks to require the C Programming Language, the faster development cycle without compiling the code and a big amount of already implemented data structures motivated the use of this alternative. On the other side, the speed of the interpreted code in python is slower then a compiled version; but in the case there is a time critical part, python allows C as a fallback programming language. Finally libraries for socket access, packet manipulation etc. are available and gumstix supports python in its buildroot[7].

[5] http://www.kernel.org
[6] http://www.python.org
[7] http://docwiki.gumstix.org/Buildroot

# 3 Network

After the hardware and basic software platform decision was made the next step was to get known the hard- and software platform, to reimplement or move the old features and to make some performance comparison.

The old relay system consisted mostly out of two parts: deployment algorithm and end-to-end communication. Both of these parts are described in this chapter and also a brief implementation overview is given. The full implementation of the deployment algorithm can be downloaded[8].

## 3.1 Deployment Algorithm

### 3.1.1 General Idea

The deployment algorithm is the part which decides whether a node needs to be deployed or not and it is quite straight forward. The node which will be deployed as the next one starts to send **probing** messages every $\Delta$ seconds. Any deployed node in range responds with a reply message containing the received signal-to-noise ratio (SNR) value of the probing message. The probing node collects all answers, records the SNR value of the reply and pushes everything into a filter.

The **filter** decides about the connection quality to the network by maintaining a table of all neighbors. The link quality to each neighbor in both directions is averaged over not more than the last $N$ values. The quality to the neighbor ($q_n$) is defined as the smaller of those two averages. Finally the filter ouput depends on the actual quality interval.

| **Quality** | **Range** |
|---|---|
| good | $q_n \geq thr_{high}$ |
| bad | $thr_{high} > q_n \geq thr_{low}$ |
| disconnected | $thr_{low} > q_n$ |

If there is at least one good neighbor link, then the filter output shows that the connectivity to the network is good. The filter indicates a "node drop" when there is no good link left but at least one bad and in every case else it reports disconnected.

Interference or going unexpected fast out of range results in missing packets. Missing packets do not update the filter and this motivates to expire the received SNR values. If the filter does not receive any message for a time interval $\delta$ it replaces the oldest SNR value for a neighbor with a dummy SNR value $S_d$.

Unlike the Mica2 integration a new enhancement is to take into account SNR values insteed of received signal strength (RSSI) and measurements in both and not just in one direction. The reason to use the link quality in both direction is rooted in the use of different types of wireless cards. This method also manages differences between the same type of cards and makes the algorithm more robust.

[8] http://wapf.ch/semestethesis07/code.zip

### 3.1.2 Implementation Details

Out of this description the requirements were getting clear. The deployment algorithm needs to be able to send raw packets as probing or reply messages; these messages should be sent as broadcast messages. The second requirement is that 802.11 wireless cards must give some indication about the SNR.

In the following part it is briefly discussed how to achieve this with the chosen platform. For this some background information about the cards and the Linux network stack are introduced.

**802.11 support** The 802.11[9] standard first appeared in the 1990s and was developed by the Institute of Electrical and Electronics Engineers (IEEE). The operational frequency of 802.11b/g is 2.4 GHz and the maximum data rate for b is 11 Mbit/s and for g 54 Mbit/s. The lifeline system is fixed at 2 Mbits/s to eliminate problems in the deployment algorithm by switching between different bit rates. The standard is mostly used in connection with an access point system. Clients connect to a central access point which routes the traffic to and from a wired connection. The 802.11b/g standard also defines an ad hoc operation mode. Each node can talk with all the other nodes in range in a peer-to-peer fashion without involving a central access points. The preconditions are that the cards are put into the ad hoc mode, they are all on the same channel and have a common service set identifier (SSID). The SSID is a code attached to all packets on a wireless network to identify each packet as part of that network. 802.11a/b supports sending packets either in a broadcast or in a unicast manner. In the broadcast form the packet is sent once and contains the broadcast MAC address (FF:FF:FF:FF:FF:FF); in the unicast form the packet is just received by the node indicated through the MAC address.

**Linux Network Access** The *Linux Wireless Tools* [10] give access to the wireless card configuration. The most important tool for this project is the *iwconfig(8)* tool which allows to put the card into the ad hoc mode, to choose the channel, bit rate and the ESSID with the following command (adapted to the project specific use):

```
# iwconfig node mode ad-hoc channel 1 essid ll rate 2M
```

After this configuration the interface is configured to send and receive raw ethernet packets from the user space to the wireless card over a Linux socket. A simple example about how to send a raw packet in python over a Linux socket is available in the channel prober code. For basic information about socket programming the information in the "Beej's Guides"[11] are helpful.

[9] http://www.ieee802.org/11/
[10] http://www.hpl.hp.com/personal/Jean_Tourrilhes/Linux/Tools.html
[11] http://beej.us/guide/

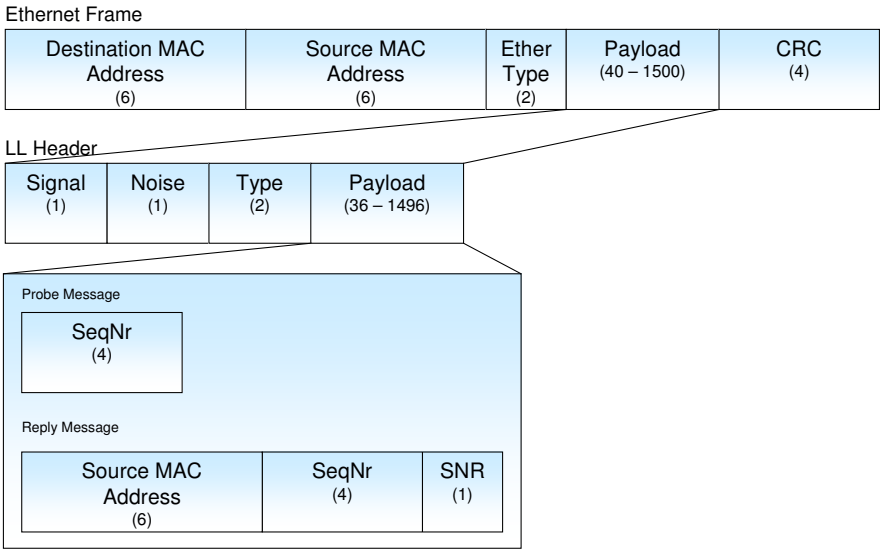



Figure 5: Ethernet - LL - Probe - Reply Message Types

**SNR Measurements** The *Linux Wireless Tools* also include a tool with the name *iwspy(8)* which allows to show the signal and the noise of the last massage received from a neighbor. The problem with this method is that not all the wireless cards have implemented a support for *iwspy(8)*, especially not the wifistix driver. The other problem with this approach is that it would be needed to poll for the values by calling regularly the *iwspy(8)* tool interface and this could cause some missed quality updates if the poll is done too slowly.

A very simple solution is to make a cross layer hack, as seen in the code listing 6. A change to the wireless driver makes possible to copy the link quality information into packets traveling all the way up through the network stack and the open socket to the user space. To copy the values into received packets, there must be a free space of two bytes (one for signal and one for the noise). The problem with this approach is that ethernet, IP and also other traffic do not have this space. A new defined LL message on top of an ethernet frame solves the problem. This message has a field for the signal and the noise and is attached to the ethernet frame with the protocol type number 1600.

In order that the receiver does not overwrite other package data, it is necessary that the transmitter already reserves this space. Figure 5 shows an ethernet packet with an ll, probing and acknowledge message types.

An alternative way to use the SNR measurements of all traffic would be to implement an extra kernel interface to get the SNR values together with the source address. The implementation complexity would be much higher.

Listing 1: wlan/wlan_rx.c

```
286 // cross-layer hack
287 if(ntohs(skb->protocol) == 1600)
288 {
289     u8* rx = (u8*) skb->mac.raw;
290     // signal
291     rx[14] = pRxPD->SNR + pRxPD->NF;
292     // noise
293     rx[15] = pRxPD->NF;
294 }
```

Figure 6: Cross Layer Hack for SNR values

### 3.1.3 Parameter Evaluation

In this section the parameters for the probing algorithm are summarized and a short explanation for each chosen values is given:

- **Probing Interval** $\Delta$ ($100ms$)
  This interval defines how often probing messages are broadcasted to the neighbors. As more messages are sent as more accurate the link quality information but also as higher the overhead. $\Delta$ is set to $100ms$ at the moment, which seems to be a reasonable value during movement. In a future implementation a probing message interval should be dependent on the moving speed.

- **Dummy Interval** $\delta$ ($=\Delta$)
  The filter expects every $\delta$ seconds a probing message from each source in the source table else, it starts to fill the filter of the appropriate source with dummy values. The value should not be shorter then the probing interval $\Delta$ else the filter would insert unnecessary dummy values. A greater value means the filter is less responsive to missed probing message. It seems setting it to the same value as $\Delta$ makes a lot of sense.

- **Dummy Value** $S_d$ (10dB)
  The dummy value is the SNR value which is insert into the filter after $\Delta$ seconds. If the node does not receive probing messages for a long time, the filter should indicate a bad link quality. This is just possible as long as the dummy value is below the bad link quality threshold. As deeper as the value is, as faster the value goes below the threshold. The value is set to 10dB since below this point the package loss rate is 100 percent according to [11].

- **Filter Window Length** $N$ (20)
  $N$ is the filter length over which the SNR values are averaged. A long filter means past received messages make the filter smoother and a message

that has just been received has less influence. It makes also the filter less responsive. The size of 20 messages is copied from the old work and it means that with an excepted receive interval $\delta = 100ms$ the values are averaged over $2s$.

- **Dropping Point Threshold** $thr_{high}$ (25dB)
  This threshold is the dropping point. As soon as the link quality to each neighbor is below $thr_{high}$, a node should be deployed. The value is based on the cut off point where the packet loss rate is 100 percent and additional dBs are added to counter multipath fading and the loss during dropping. Evaluation showed that a value of 25dB is a good choice not to loose network connectivity. In some cases it might be better to put it to a less restrictive value.

- **Bad Link Quality Threshold** $thr_{low}$ (20dB)
  With this threshold it is possible to recognize a big decrease of the link quality during a drop. If the node quality goes under the threshold the filter indicates a bad link and the mobile user might use local placement(3.1.5) to adjust the node's position until it is not anymore in a bad fading area.

### 3.1.4 Controlling and Monitoring

The gumstix computer does not have any display or input capabilities implemented. To indicate the filter status and to start and stop the node, it is necessary to have a kind of input and output. A special breakout board[12] for the gumstix allows to attach different screens or LEDs and also buttons; unfortunately the actual implementation uses just the mobile user system to monitor the filter status and to start and stop the nodes remotely. A small script starts an XMLRPC server[13] and draws a PyGTK[14] window on the mobile user system side. Through remote procedure calls the node can show a text and a background color on the PyGTK window. On the node side another XMLRPC server is running after the boot procedure and makes it possible to change the state of the node over the wireless network beween doing nothing, acting as a relay or as the probing node.

### 3.1.5 Local Placement

The Mica2 system has the problem that the signal strength can vary up to about 20dB over distances in the order of 10cm due to local multi-path fading. It avoids the placement in a local deep fade with a LED indication of the nodes link quality. The new system does not need such a special LED indication of this case because the node that will be deployed as next one is the measuring one and not the mobile user system. If local placement is desired so it is possible to move the node

[12] http://docwiki.gumstix.org/Expansions#breakout-gs
[13] http://docs.python.org/lib/module-SimpleXMLRPCServer.html
[14] http://www.pygtk.org

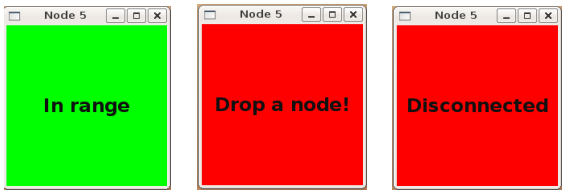



Figure 7: Filter Quality Level Monitor

after the deployment until the quality monitor indicates an appropriate link quality (Good or Drop a node).

### 3.1.6 Power Control

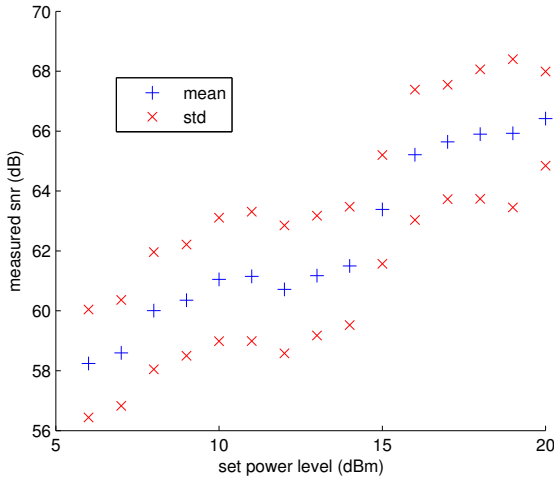



Figure 8: Measured SNR Values with a NetGear Card sent at different Power Levels by a wifistix

Power adaption is not yet implemented into the new 802.11 prototype but according to the measurements in figure 8 it is possible to use it. The figure shows the averaged received SNR value while probing every 100ms during 100s for all available power levels. Although the measured values are not monotonic growing the differences are still usable. An evaluation and deeper analysis of the 802.11 automatic power adjustment could maybe already do its job but could also influence the probing algorithm. For this reason the relays with the wifistix are fixed at a specified transmission power rate of 18dBm. To change the transmission power calls to *iwpriv(8)* and *iwconfig(8)* of the *Linux Wirless Tools* are necessary.

```
# iwpriv node powercfg 0 0 0 0
# iwconfig node txpower 18dBm
```

The compact flash cards do not support dBm values as an input and just differentiate discrete values between 0 and 255. The calls are different but the transmission power is also fixed for now:

```
# iwpriv node alc 0
# iwpriv node writemif 62 128
```

The first command of these two configuration calls respectively turns off the auto power management and the second sets the power level.

## 3.2 Routing Protocol

### 3.2.1 Optimized Link State Routing Protocol - OLSR

The Optimized Link State Routing Protocol[15] is a routing protocol that is optimized for mobile ad-hoc networks. It is a proactive link-state routing protocol that floods a full topology table to all nodes in the network which then compute optimal forwarding paths locally. The lifeline system uses after an evaluation of two other routing protocols (AODV and DYMO) a user space olsrd implementation[16]. This version also implements an optional link quality extension.

The main reason why OLSR is used in the system is because it is a well tested and widely used linux user space implementation (i.e. it is easy to compile it for different platforms). Even if it makes the routing a bit slower compared to the kernel implementations of AODV[17] or DYMO[18], it does not affect the kernel stability and it has already a useful link quality extension with a lot of output information.

The **link quality extension** is based on a Expected Transmission Count (ETX) measurement. The extension optimizes the path to the destination by choosing the one with lowest number of necessary transmissions. ETX is a value which is calculated based on received HELLO messages from all the neighbors sent regularly. If a node receives $n$ packets from a neighbor out of $N$ then the probability $\frac{n}{N}$ is called Link Quality ($LQ$) to the neighbor. It indicates how likely it is that a packet sent is successfully received by the neighbor. Unicast traffic includes also acknowledgments to our sent packets. The ETX says how many retransmissions are needed to get the message through. A message is retransmitted when either the message itself is getting lost or the acknowledge. In order to calculate the ETX it is necessary to know the link quality in the other direction also called

[15] http://tools.ietf.org/html/rfc3626
[16] http://www.olsr.org
[17] http://w3.antd.nist.gov/wctg/aodv_kernel/
[18] http://ianchak.com/dymo/

Neighbor Link Quality ($NLQ$) which is transmitted with the HELLO messages. The probability that we need to retransmit a packet is $LQ * NLQ$. $\frac{1}{LQ*NLQ}$ is the average number of transmission attempts needed until the message gets through to the neighbor. Finally the optimal route is the one with the minimum ETX accumulated over all hops.

### 3.2.2 OLSR Configuration

The used OLSR implementation allows to configurate the system through a configuration file. The most important changes to the standard file are the activation of the link quality and http output extension as well as the interface settings.

The following parameters are important for the link quality extension. Because of the special environment with the high mobility the values are adapted accordingly. It seems that the default values are optimized for a static network.

An interval of two seconds between two Hello Messages seems to be too long. At the time of the deployment of the actual probing node it is very likely that the mobile user system still routes directly over the previous hop because it minimizes the end to end ETX. After the deployment, while the mobile user system is walking away, the actual route is getting worse and it will change soon to new deployed node. It takes several hello messages to decrease the ETX enough to change the route. Assumed the mobile user system moves within 1.5m/s (like a fire fighter) it takes several meters until the actual route is recognized as bad. It could be that the system is already out of range and causing interrupts in real time applications. The system seems to work fine with a hello message every half a second.

The link quality window size is also coupled with this problem. As higher the window size is as longer it takes to gradually reduce the ETX to zero. But longer window sizes have the advantages to be more robust against temporary influences. With a value of twenty the history is ten seconds which seemed to be a good value especially because of the possibility to define a hello validity time.

The HelloValidityTime defines after how long a neighbor should remove a sender from its list as an active node. The value is sent within the hello messages. It solves a part of the above described problem. With a validity time of three second a person walks in the average 4.5 meters which should be considered by the SNR deployment threshold.

These values are not optimized and further investigations are necessary to improve the route change quality. More information about the routing protocol and the link quality extension can be found under [3], [12].

| Parameter | Description | Default | Changed |
|---|---|---|---|
| HelloInterval | Interval between two HELLO messages | 2s | 0.5s |
| LinkQualityWinSize | History of the last $N$ messages | 10 | 20 |
| HelloValidityTime | Node's expiration time | 20s | 3s |

# 4 Evaluation

In this section results of throughput and roundtrip time measurements are presented. The measurments are based on a network build out of the mobile user system, nine relay nodes and the base system.

## 4.1 Throughput

The throughput is measured over a TCP stream with the *iperf(1)* tool[19]. This tool allows to measure the throughput over TCP and UDP. Problems with UDP streams at higher data rates with a big packet size forced us to use a TCP stream. According to the survey in [10] TCP streams tend to have performance issues in wireless networks and the actual throughput can highly vary. However it is still interesting to know what is possible over a deployed multihop network and the measured values should more be seen as lower boundaries especially since the system is running at the low fixed bitrate of 2Mbits/s.

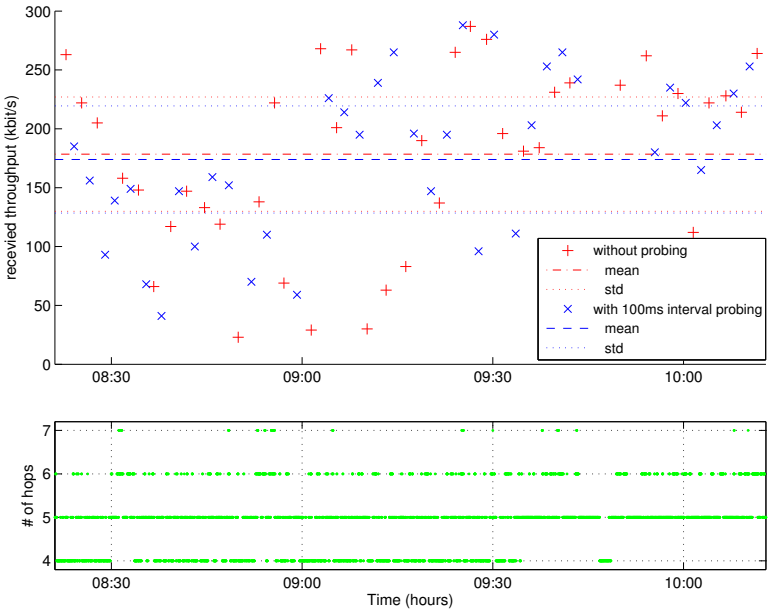



Figure 9: TCP Throughput with OLSR and its Standard Parameters

Figure 9 shows multiple measurements of the throughput averaged over 60 seconds while OLSR was running with a standard configuration. The test switched

[19] http://dast.nlanr.net/Projects/Iperf/

always from one to the other trial between with and without probing on the mobile user system side. All the nodes were running with minimal transmission power and the dropping algorithm parameters mentioned earlier in this report in section 3.1.3 were used. Each deployed node was in the rather restrictive "drop node" quality range which resulted in more interference with neighbors.

The variety is expected high and some of the measurements are as low as 23kbit/s but on the other side there are high throughput measurments with a peak of 287 kbit/s which is more then 14 percent of the 2Mbit/s raw bit rate over a network with a changing number of hops between 5 and 7. Because UDP is not acknowledged and does not have all the overhead of TCP, a higher throughput should be possible and it might be even a more stable one as long as the network is not overloaded. UDP is mainly interesting in the area of realtime protocols like audio and video. According to this measurements the probing algorithm origins just a small throughput decrease.

A second trial a couple of hours later showed a slightly different picture. The number of hops was compared to the first trial never below 5 although the nodes were positioned at the same locations in the same order. The bandwidth was not as high as before but the variety decreased.

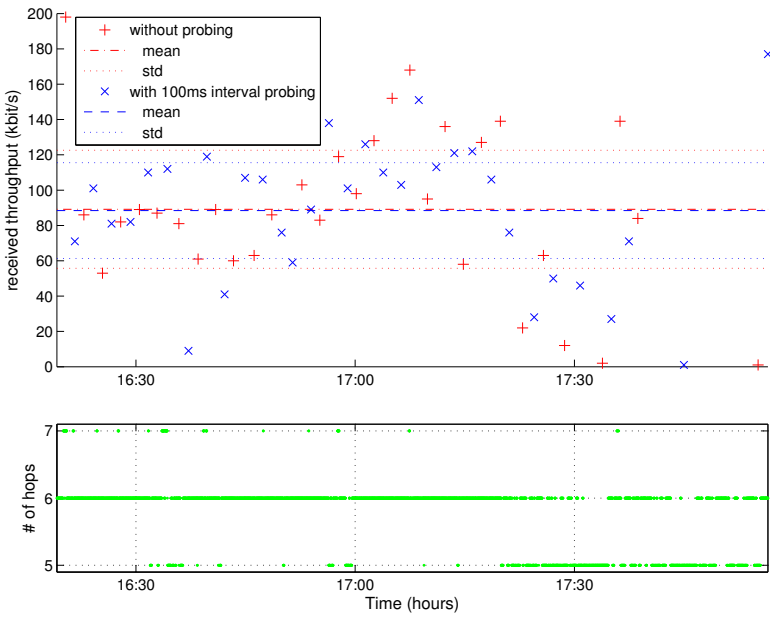



Figure 10: TCP Throughput with OLSR and its Standard Parameters

## 4.2 Roundtrip Time

The round trip time is of interest for realtime protocols and shows the delay from one end to the other end of the network. With the same configuration and the same positions for the nodes as before the first step was determining the saturation point of UDP traffic. UDP does not include congestion control and allows to over-utililze a network. While the network's utilization is very high the RTT is disproportionately high. It is more interesting to measure the round trip time while the traffic is as high as possible but without a big loss of packages. To make the measurments more interesting the packet size of the traffic is set to 64 Bytes which is the same as the traffic of the real time protocol in our system.

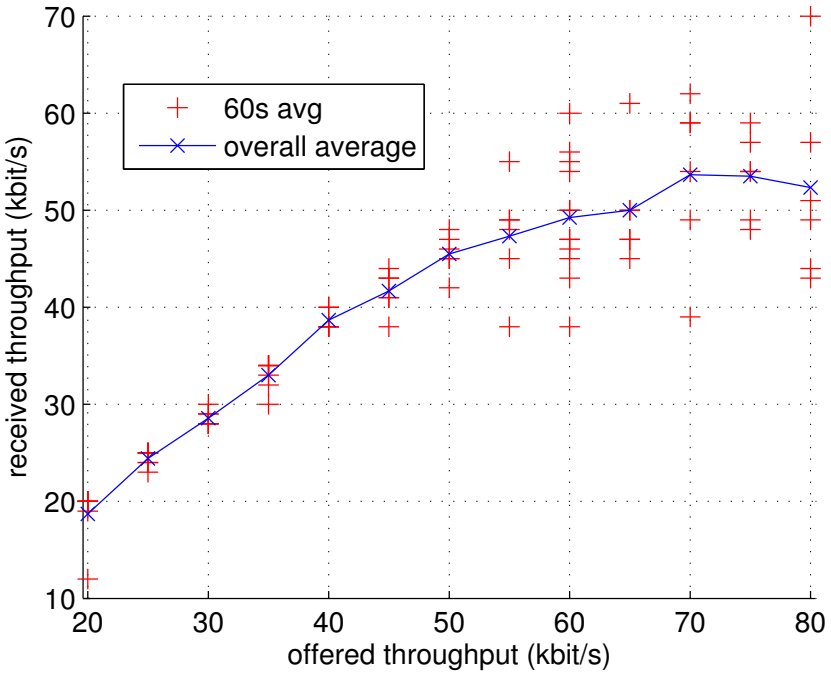



Figure 11: Throughput of 64Bytes UDP Packets

Figure 11 shows the offered load which is sent by the base system to the linux network stack versus the relativley low throughputs due to high overhead of small packets recorded on the mobile user system's side. Again the measurements are taken each time over 60 seconds. The network does not seem to have any problems with traffic below 50kbit/s. Starting with the offered load of 50Kbit-s/s the difference between the maximum and minimum received traffic increases. 50Kbits/s and 60Kbits/s were the point for the round trip time investigations and the results are presented in figure 12. UDP throughputs up to 250 kbits/s were observed using large packets, but the system regulary crashed when operated as

such, possibly due to incorrect internal handling of buffer overflows.

For both saturation points multiple measurements over 60 seconds are taken. The pings for the RTT measurments are sent every second and finally averaged over the single test interval. The RTT is good as long as the received throughput is close to the offered throughput. If there is congestion in the network the RTT increases very fast to high values. The minimum for the offered 50kbits/s is 22.27ms and the maximum 1009ms with an average of 107.5 ms. For the offered 60Kbits/s the average is 190.8, the minimum 24.16ms and the maximum 2400ms.

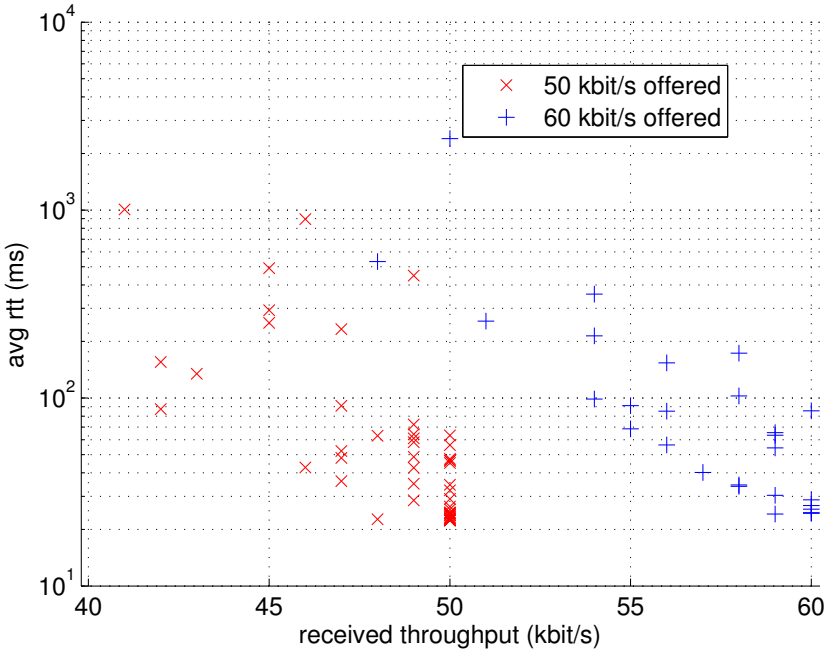



Figure 12: Round Trip Time (RTT) around the 64KB UDP Packet Throughput Saturation Point

# 5 Applications

Based on the deployment algorithm, it is now possible to build a communication network in real time. The last section showed the bandwidth is much higher compared to the old 38.4Kbits/s capable Mica2 nodes. Finally the next step is to use the network with applications whereas the IP stack supports that each node can talk with all others in the network. The introduction described that one target of this work is to add the basic infrastructure for multiple first responders. This target is hereby reached.

In the following sections some possible useful tools for the daily life of a first responder like a firefighter are described.

## 5.1 Sensor Values and Localisation

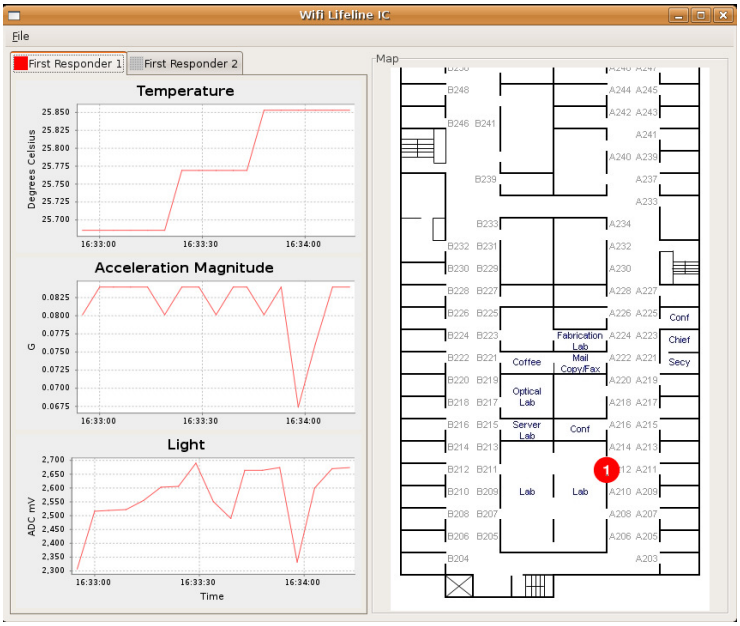



Figure 13: Example Incident Commander GUI

As the old system, the new one supports sensor values and localisation based on a RFID reader at the mobile user system. The actual implementation uses a temperature and light sensors and for the movement indication an x-/y- accelerometer.

Because the x-/y- accelerometer does not indicate the speed and it is not very accurate, the magnitude of the x and y acceleration is an indicator for the actual movement. If the magnitude goes over a specified threshold then the mobile user is moving else motionless.

The sensor values and the RFID tag information are accessible through an XMLRPC server on the mobile user system. Any node in the network can register itself for the values through an RPC call. Each time the mobile user system scans a tag or there is a new sensor value available, the values will be published to all subscribed listeners. The publishing mechanism is also handled by an RPC call.

## 5.2 Text Message, VoIP & Video Software

Linphone[20] and ekiga[21] are two voice over ip (VoIP) softphones. Both of them support different kind of codecs for voice and video coding and have an instant messaging feature to send text messages back and forth. These soft phones use the Session Initial Protocol[22] (SIP) which allows to make peer to peer message exchange possible. While walking away from the base system some short breaks up to a few seconds are now and then recognizable during the switches from one hope to the other. The overall voice quality is pretty good and with further optimization it it likely that one could eliminate almost all of these interruptions. After the network is deployed but sill while walking around such hissing is very seldom. For video applications it is a trade off between the number of frames per time unit and the frame quality. The transfered pictures are not of very high resolutions but it is still possible to get an impression of the environment of the mobile user system. In some applications it might be enough to have just every several seconds a new picture or even just on request.

[20] http://www.linphone.org
[21] http://ekiga.org
[22] http://www.sipknowledge.com/SIP_RFC.htm

# 6 Summary and Future Work

Starting with the hardware and basic software selection a plattform evolved making it possible to design and programm a dropping algorithm which allows to build a multihop network with 802.11b/g wireless cards with the ad hoc mode in realtime. The network capabilities increased through evaluating, adding and configurating the OLSR routing protocol with its link quality extension and allowed to make basic measurments of throughput and end-to-end delay. The measurements showed that a lot of different applications like audio und low quality video transmissions are runnable on top of the network but also that some limitations exist. Finally the localisation and sensor value measurement parts were ported to this new platform to demonstrate the system in an example fire fighter scenario.

During the work on this project many ideas for extensions showed up and the system is not highly optimized yet. The following list summarizes points for the main future improvements:

- As already mentioned in the section 3.1.6, the 802.11b/g wireless cards allows to adjust the transmission power. A detailed analysis of the power adaption possibility of such cards and a introduction of a power adjustment algorithm could improve the run time, reduce interference and help to improve node to node link quality after a change in the environment.
- The influence of different bitrates to the probing algorithm is unknown but a higher throughput is expected.
- The routing protocol is not entirely optimized. Just the most important parameters are adjusted to meaningful values. More measurements and evaluations could confirm either the chosen parameters or would it make easier to set better values. The routing protocol does not include the SNR measurements and the speed of a mobile user system. It should be possible to decrease the Hello Message interval in static areas and increase it in areas with mobile users systems. The accelerometer of the sensor board could be used to indicate the movements.
- The dropping algorithm uses the SNR values from the probing and acknowledge messages only. As already mentioned it is possible to reuse the signal and noise values of all packets received by the wireless card. As a result it wouldn't be necessary anymore to generate extra traffic to measure the link quality to the neighbors all the time and the overhead would be decreased.
- The thresholds of the dropping algorithm are not evaluated very accurate right now. Different scenarios like going around a corner, through different kind of doors or walking in a stairwell seem to influence the SNR values in different patterns. Further investigations could increase the range and the flexibility of the system.

- At the moment the relay nodes just build a network and forward the traffic to another node. Multi casting could be one way to reduce the used bandwidth to send information to multiple mobile user systems. In the case of audio streaming making nodes smarter and allow to mix audio signals together could decrease the used bandwidth in group chat environments.
- Not all traffic is of the same importance. A short break in the video transmission might be much more acceptable than some missed couple of seconds of an audio transmission. The probing messages should have higher priority than the application traffic because they are responsible for the network connectivity which is used by the upper layers for a proper work.

# References


[1] The connex motherboard. http://docwiki.gumstix.org/Basix_and_connex.

[2] gumstix inc. http://gumstix.com/.

[3] Olsr implementation. http://www.olsr.org/.

[4] wifistix daughterboard. http://docwiki.gumstix.org/Expansions#wifistix-FCC_and_wifistix-EU.

[5] Lipo charger/ max 1555 breakout board. Technical report, SparkFun Electronics, 2006.

[6] U. Batteries. Ubc 433475 - technical datasheet. http://www.ultralifebatteries.com/documents/techsheets/UBI-5122_UBC433475.pdf, July 2006.

[7] Crossbow. *MTS/MDA Sensor Board Users Manual*, June 2007. http://www.xbow.com/Support/Support_pdf_files/MTS-MDA_Series_Users_Manual.pdf.

[8] J. Geissbühler. Wireless multihop communications for first responder connectivity. Master's thesis, Eidgenössische Technische Hochschule Zürich, 2006.

[9] A. I. T. GmbH. Acg dual iso cf card reader module rdhp-0406p0-01. Technical report, ACG Identification Technologies GmbH, 2005. http://www.tagtechnology.it/download/CFReader13,56MHzDualISO.pdf.

[10] V. O. Ka-Cheong Leung. Transmission control protocol (tcp) in wireless networks: Issues, approaches, and challenges. *IEEE Communications Surveys & Tutorials*, 8(4):64, 2006.

[11] M. R. Souryal, L. Klein-Berndt, L. E. Miller, and N. Moayeri. Link assessment in an indoor 802.11 network. In *Wireless Communications and Networking Conference, 2006. WCNC 2006. IEEE*, volume 3, pages 1402–1407, 2006.

[12] tlopatic. olsrd link quality extensions. http://www.olsr.org/docs/README-Link-Quality.html, 12 2004.