Data Traffic on Mobile Applications and its Impact on Battery Life Time

Kai B. Heinz¹

Abstract
The energy consumption of mobile devices, such as smartphones and tablet-PCs is a major concern. In this paper a sample application will be analyzed to determine how much energy can be saved by reducing the traffic sent in a wireless network. The application is called ArtNet-DMX-Controller and is built by the author himself. It is an Android application to control stage device such as lamps, beamers, moving heads … For sending of data to the exact stage devices a special hardware infrastructure is needed. The backbone of this setup is a star-based computer network. The tablet running the application is connected to this network by Wi-Fi. By reducing the amount of data send to the network lower energy consumption is assumed. To prove this assumption two versions of the app had been tested with the same settings. The old version used to send always, whereas the improved version sends data only when necessary.

Chapters 1 and 2 are of fundamental interest, describing why it seems to be important to reduce traffic and who did similar work. Chapter 3 and 4 are describing the application, so that everybody can understand basic principles of the app, which is from utmost interest when interpreting the test results. Chapter 5 alights the measurement and shows the test results, which will be concluded in chapter 6. Chapter 7 shows further improvements to the application and measuring systems.

1. Motivation
Mobile applications are a new field, for users as well as for software developers. Due to the sophisticated technology mobile devices use, it seems that mobile applications can be developed just as usual computer programs, but the restrictions a mobile device has must be always be in mind. One major issue is that mobile devices are not stationary and therefore usually does not have a constant power source. Commonly they are powered by rechargeable batteries. How long a battery can be used without recharging depends on multiple factors. One is the capacity of the battery and another one the amount of energy stored in before it had been disconnected from the charger (In other words: if it is used before fully loaded). Another impact has the software installeed on the mobile device. Regarding to Wilke (Wilke 2012) it is possible that different application use a different amount of energy, even if it looks to a user as they are doing the same. This is why we have to take a deeper look into the applications.

For this paper this had been done by on one of the authors own applications (see chapter 3) and saw that the source code can be changed without changing the function or usability of the program. The idea was to reduce the wireless data traffic to extend the life time of the battery (By talking of the battery life time, the time between two charging periods is meant). Regarding to Cisco (Cisco 2011) the data transmitted by mobile devices will increase 26 times between 2010 and 2015, reaching 6.3 EB per month. Even if the ArtNet-DMX-Controller works only in a local network and is not connected to the World Wide Web, the impact of the changes to the source code can be measured and used as an entry point for further reduction of wireless data traffic.

However, the extension of the battery life time has more than one effect. Because of less charging cycles, the total life time of the battery grows. That is especially important in modern devices where the battery cannot be removed by the user himself. Another effect is that the user can use the mobile device

¹ University of Applied Sciences/Trier, Umwelt-Campus Birkenfeld, PO Box 1380, D-55761 Birkenfeld, Germany, mail@kaiheinz.de, http://www.kaiheinz.de
longer without charging, so the use experience grows. And of course the energy consumption is going down, what leads to a greater sustainability.

2. Related Work

Energy consumption of software is often ignored by developers. But in the field of mobile devices energy consumption is crucial. Much research had been made in the last years, setting up measuring environments and comparing applications.

Dick (Dick 2012) compared how websites can be designed to minimize the data traffic. By compressing images and using HTML and CSS instead of text inside images, using caches, they could reduce the traffic needed to display the website to a user. This work inspired my own research in this field. Hence after they did all these improvements, they had to use some kind of measurement circuit to proof their work. This measurement setup built the basis for my own measurements. However, they measured personal computers and servers, whereas the ArtNet-DMX-Controller is a tablet-PC application which has some other aspects to take care of.

Wilke (Wilke 2012) are thinking of a labeling system for smartphone/tablet applications to enable a user to decide on a proved energy consumption rating and not just on the five-star rating as Google Play is using it until now. To introduce such a labeling, they also defined a measurement environment. To simulate a user they use Android test cases, which will be brought to phone by USB. The energy consumption of the Device under Test (DUT) is then monitored by two energy-meters, one measuring the energy brought to the device by USB, the other one checking the energy consumption between device and battery. A similar setup had been used to monitor the energy consumption of the ArtNet-DMX-Controller. However, the main focus on this work lay in the relation between data traffic and battery life time.

3. About the “ArtNet-DMX-Controller”

The ArtNet-DMX-Controller was developed between September 2012 and May 2013 based upon the idea to control stage devices such as lamps, beamers, scanners, fog machines and similar others. One requirement for such an application was that it has a standard DMX interface. DMX is a very popular bus-system used in stagecraft and can be seen as a de-facto standard. This way the ArtNet-DMX-Controller can be used with existing hardware and only minor changes to the hardware infrastructure. Another important requirement was the User Interface, which should be simple to use, but has all the features of an existing light console.

Die Architektur soll zum Ende des Projektes durch die Schaffung eines „Konfigurators“ komplettiert werden, welcher im Webportal die Suche nach Schlagwörtern (bspw. Ökobilanzierung, Energiedatenerfassung) ermöglicht und so den richtigen Installer, d.h. den OpenResKit HUB mit den benötigten Domänen und die jeweiligen Desktop-Clients, generiert und somit für die persönlichen Bedürfnisse des Anwenders das notwendige Softwareportfolio bereitstellt.

Figure 1: Hardware Infrastructure
Figure 1 shows a basic setup. On the left image a tablet-PC with the application installed is shown. The application adapts a common light desk to allow users an easy transfer from existing solutions. The center image highlights the backbone of the infrastructure. Both devices, a wireless router and an ArtNet-to-DMX converter, are easily to get and, considering the costs of the whole construction are fairly cheap. They are connected with a common Ethernet cable. The lamp in the right image indicates all the devices that shall be controlled. Each lamp, scanner, beamer, etc. needs to have a DMX-interface. Because this is very common today, no additional costs arise at this point. To have best control, each device must be programmed with a unique address, however smart addressing allows a stage technician to save several channels. The connection of the stage devices is done with a specific DMX-cable.

The hardware structure leads to a first overview of how the application works. The ArtNet-DMX-Controller sends a package with DMX-data via Wi-Fi to the router, which sends it further to the converter, which unwraps the data and feeds the stage devices with the received information. The package itself uses the ArtNet-Protocol. By using this open protocol any ArtNet-to-DMX-converter can be used. In the future it might be possible that stage devices can directly interpret the ArtNet-protocol, which would make it possible to use the application directly without any converter.

A User can control stage devices with the ArtNet-DMX-Controller in two Modes. These Modes apply to two different areas of the application UI. Both areas contain multiple grouped faders (so called seekbars). In context of this application one group of faders is called a Bank. The following will describe the Modes performed by the Banks:

**Direct Mode:** For controlling the channels directly a value for the channel can be set in Bank A. The values will be collected and sent to the stage devices.

**Chasing Mode:** The chasing Mode allows the User to ‘program’ a chaser with a sequence of light scenes, which will be sent to the DMX-devices independently. For programming a User has to setup the scenes in Bank A and store it on the chaser (a fader in Bank B) he wants. The User can also define the speed of the chaser and the fading time. The range of the speed can be set from 0.3 seconds to ~10 minutes. To start a chaser, the fader just has to set to any position ≠ 0. The value of the fader serves as a master for the specific chaser.

Beside that there are master faders for Bank A and B; fade in/out, black out and flash buttons are available. These functions are not relevant for this paper and therefore not described deeper.

From a software point of view, the values in Bank A are simply sent to the ArtNet-Library, which is responsible for sending the values. Bank B is truly more difficult. Each chaser has a list where the light scenes are stored. By programming a specific chaser, this list of scenes will be modified/extended. For each chaser exists a thread, that will be responsible for sending the scenes. This way each chaser can have its own speed and fading settings. However, it is possible that two chasers want to set a value for the same channel. In this case the highest-value-first principle is used. This had been implemented with a max-heap for every DMX-channel. If a value changes, its position in the heap will be updated. The highest value will always be in the root. The actual sending of the data is done in the so called ArtNet-thread. This was the point where the improvements took place.

4. Sending Data

Sending data to the ArtNet-DMX-Converter is the keystone of the application. It is performed within the ArtNet-Library. There are only a few methods collected in the ArtNetFacade-class to communicate with the library. Within this class the described max-heap is used to store values for all channels. The advantage of a max-heap is that the retrieval of the highest value can be performed in $O(1)$ constant time and all other operations such as inserting and updating have also a very good $O(\log n)$ time.
The ArtNetFacade contains the ArtNet-thread which is responsible for sending the data. Within the thread an endless loop collects data and sends them out to the stage devices.

4.1. Continues sending

Figure 2 shows how the application used to work. Values, either entered from a User or from one of the chaser-threads, are sent to the ArtNetFacade where they get stored. These values can arrive always and they are totally independent from any other event. Especially the threads are working autonomously and with many different settings for speed and fading. To make sure that the stage devices will always receive the latest changes, they will be updated regularly every 30ms.

```
1 WHILE TRUE
2  DO BEGIN
3    FOR i := 0 TO 511
4    DO x[i] := getMax(i);
5    sendValues(x);
6    wait(30);
7  END;
```

Listing 1: The ArtNet-thread as it used to be

4.2. Improvements

The changes affect the waiting time of the ArtNet-thread. Obviously the longer the waiting time is, the less values are sent. Regarding to (Dick 2012) the less data sent over a network, the less energy will be used. In practical scenarios, a chaser will be set to a speed-time somewhere between 2 seconds and 10 minutes. This leads theoretically to a decrease of traffic between a factor of 66 and 20,000, depending on the actual settings. These factors are so great, that a difference in battery life time should be observable, which would lead to a longer battery life time and a better user experience.

Figure 2: Sending Data received from the User or a Chaser
To extend the waiting time, another waiting method had been inserted. It is not coupled to any time, so that it could pause the thread forever. To avoid this, the thread will be notified immediately after a new value from a user or chaser arrives. In fact these changes are just a few lines of code, but if the theory works out, a real difference can be measured.

```
1 WHILE TRUE
2   DO BEGIN
3      WAIT(∞);
4      FOR i := 0 TO 511
5         DO x[i] := getMax(i);
6      sendValues(x);
7      WAIT(30);
8   END;
```

Listing 2: The improved ArtNet-thread and the related setValues-Method

One may ask why there are two waiting methods now, instead of a single WAIT(∞) method. This has practical reasons. If there are too many data send it could lead to buffer overflows in the ArtNet-to-DMX-Converter, because DMX has a specified minimum sending interval. In practice the NOTIFY(), can only release the WAIT(∞) method, so that this solution can never send faster than the one in listing 1.

5. Measurement

5.1. Measurement Method

Measuring an energy consumer can be quite simple nowadays. By putting an energy monitor into a wall plug, a customer can easily see how many energy had been consumed by this device in a period of time. Measuring software is much more difficult, because modern programs always rely on an OS, which by itself needs energy. To get rid of this overhead two measurements can be done, one with the application running and one without. The delta can be seen as the energy consumed by the application. Another way is to ignore the energy used by the OS and comparing applications directly. Because both applications need the OS in the same way, the results are good enough to compare. Nonetheless, it cannot be seen how much energy the application exactly used, just how much more/less it uses compared to the other program.

To measure how data traffic impacts the battery life time the second approach had been used. The main interest was how much energy the application needs to operate and how changes in the app (done as described in the context of sending data) affect these values.

But measuring a mobile device can be much more difficult than measuring a PC or a fridge. Because a modern mobile device has at least two energy providers (the charger and the battery), we cannot just measure how much energy had been flown from the wall plug into the device, because we do not know how many of this electricity is used for driving the device and how much is used for loading. Using just one energy input might be the simplest answer, but modern device have often built-in batteries which makes a removal very difficult or even impossible.

For measuring the ArtNet-DMX-Controller the battery could be removed, so that all the energy measured at the charger has a direct relation to running application. The ArtNet-DMX-Controller was running on a Samsung Galaxy Tab 10.1 with no other apps running. However some OS specific applications could run in the background, but as long as they run on every test, they can be ignored. On the other end of the line two 5-channel RGB LED lamps had been used. Lamp 1 (L1) on channel 1(up to channel 5) and lamp 2 (L2) on channel 9 (up to channel 13).
5.2. Measurement Targets

The main target was to research how the data traffic affects the battery life time. The changes in the application are described above. Intuitively the battery life time should be extended, if the data traffic exceeds. To prove this theory multiple tests had been done with the old and improved version of the ArtNet-DMX-Controller. One test run was 30 minutes.

Every test had been run with the old and improved version of the ArtNet-DMX-Controller. At first the direct mode had been tested. Therefore a simple light (L1: Red, 100%, L2: Green 100%) had been set up. After that the Chasing Mode had been tested. Because of the variety of possible settings, two sample chasers had been programmed for each test run. Chaser 1 with the scenes: (Red, 100%) → (Green, 100%) → (Blue, 100%) on L1 and Chaser 2 with the scenes: (Green, 100%) → (Red, 100%) → (Blue, 100%) on L2. Both chasers had been started at the same time by hand. During the tests both chasers are active and run therefore in an endless loop. The following settings for speed and fading (both in seconds) had been used:

<table>
<thead>
<tr>
<th>Chaser (fast)</th>
<th>Chaser (medium)</th>
<th>Chaser (slow)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(speed: 3/fading: 1)</td>
<td>(speed: 30/fading: 10)</td>
<td>(speed: 600/fading: 200)</td>
</tr>
</tbody>
</table>

Table 1: Speed and Fading Values used for Testing

“speed” is the overall time of a scene, whereas “fading” is a period of time used from the overall speed to do the fading. Therefore fading ≤ speed must apply.

5.3. Measurement Results

Figure 3: Energy Consumption in Direct Mode

Comparing the results from the energy meter, a change in energy consumption can be directly observed. Figure 3 (Direct Mode) and 4 (Chasing Mode (medium)) are showing the results in a line diagram. In the diagrams the blue line is the energy used by the old application and orange indicates the improved version. Comparing the results of the Direct Mode (figure 3) two nearly constant lines can be seen. The orange line for the improved application is constantly lower than the blue for the old application. By comparing the total energy consumption (table 2) an average reduction of 0.2379 W could be computed. Figure 4 serves as an example for the three tests in Chasing Mode. While the light keeps constant, the energy consumption is constantly lower than in the old application. The explanation of these cyclic energy increases is the fading time, which had been set to 1/3 of the total speed. While the chaser is not fading, it can be handled as in Direct Mode (That is 2/3 of the time). However, while fading the energy consumption
increases due to sending data. During this time the energy consumption is on the same level as the old version was all the time.

![Figure 4: Energy Consumption in Chaser Mode Medium (30/10)](image)

The energy used by the old application is nearly constant at 6.7 W. Only in Direct Mode it is a little bit lower. This can be explained by less work for the max-heap. Because no new values arrive, no sorting is needed in the heap. In Chasing Mode new values arrive regularly, but mostly while fading. Regarding to our test scenarios, 1/3 of the total testing time is used for fading. So 1/3 of the time in every test sorting in the heaps is needed.

In both diagrams unregularly peaks and an overall floating of the lines can be seen. The peaks are usually produced by some action of the OS, i.e. trying to update date and time or checking for software updates. These peaks falsify the result, but in order that they can occur anytime and on any measurement they will be ignored. A general floating of the line is more difficult to explain. One idea is that a person was standing/walking through the WiFi-Signal, so that the OS tried to extend/reduce the transmission power. A method to improve these falsified results is to measure multiple times and using statistically methods to reduce the error.

<table>
<thead>
<tr>
<th></th>
<th>Direct</th>
<th>Fast</th>
<th>Medium</th>
<th>Slow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Old (avg)</td>
<td>6.5492 W</td>
<td>6.7078 W</td>
<td>6.6966 W</td>
<td>6.7114 W</td>
</tr>
<tr>
<td>delta</td>
<td>0.2379 W</td>
<td>0.2355 W</td>
<td>0.3409 W</td>
<td>0.3284 W</td>
</tr>
</tbody>
</table>

Table 2: Average Results of the tests

Comparing the old and improved version of the ArtNet-DMX-Controller, the improved version is always consuming less energy. However, the faster values change (see table 2 - Fast), the closer it comes to the old version. This can be explained with a look to the pseudo code in listing 2. The WAIT(∞) method in line 3 is the main point why the traffic deceeds. If a lot of values are coming fast the WAIT method actually waits for a very short period of time. This leads to a very small reduction of data traffic so that the total impact of this method is virtually zero.
To set these values in a context, the capacity of the battery needs to be considered. On the Samsung Galaxy Tab 10.1 we used, the battery capacity is 25.9 Wh. Using the old application with the settings used for the Medium test on this device a maximum running time of 3.867 h can be reached (it will be less because the OS will shut down before reaching total zero). With the improved application 4.075 h are possible. This is an extension of 12.48 min. or 5.379%.

6. Conclusion

Developing for modern mobile devices such as smartphones or tablets seems to be easy from a software developer’s point of view. The APIs are open, IDEs can be downloaded for free and High-Level-Programming is available. This can lead to the assumption that there is no need for caring about memory, traffic or energy consumption. But especially the last point is crucial on mobile devices.

In this paper it could be shown that small changes in the source code can have a big impact on the total energy needed for execution. It could further be shown that there is a direct relation between the data traffic an application produces and the energy needed.

Measuring mobile applications is still a challenge. The ArtNet-DMX-Controller could be analyzed just by removing the battery of the tablet-PC. Thus the battery is built-in the device and not removable from outside, this test series will stay unique. For a software developer it is not deemed acceptable to disassemble a mobile device every time before energy tests can be done.

7. Future Work

While testing it could be observed that the screen uses by far the most energy. When the screen was turned off, the energy consumption drops enormously. For the ArtNet-DMX-Controller it seems to be inadequate to shut off the screen. However, Roberson (Roberson 2002) showed that screens need less energy for displaying dark than for light colors. Even if this mainly refers to CRC displays, a black background will be used in the future to lower energy consumption further.

As mentioned above it deems not acceptable for a software developer to disassemble a tablet-PC before running energy tests. Thus energy consumption is such an important matter when developing for mobile devices a simpler method should be established to allow developers energy testing. Dick (Dick 2012) mentioned in another context that it would be very efficient if IDEs such as Eclipse or Netbeans could give hints on energy consumption beside other hints, i.e. language support. The work of Pathak (Pathak 2012) described how energy consumption can be approximated. This could be a starting point for developing smart energy aware IDEs / IDE-plugins.

Bibliography


