Parallel Data Transfer with Voice Calls for Energy-Efficient Mobile Services

Jukka K. Nurminen and Janne Nöyränen

Nokia Research Center, Helsinki, Finland
{jukka.k.nurminen,janne.noyranen}@nokia.com

Abstract. Battery consumption is one challenge for mobile applications and services. In this paper we explore the scenario where mobile phones delay the transfer of non-urgent data and perform the communication while a voice call is active. Our measurements show that data transfer during voice call requires only slightly over 10% additional power and that simultaneous voice call slows down the file transfer by 3%-14%. As a result we can save over 80% of energy in data transfer if we can delay the communication to a time when user is speaking at the mobile phone. For a user speaking 26 minutes a day this would allow 50MB of low energy data communication. A large class of applications can delay their data transfer without major effect to the user experience. The power saving mechanism can be implemented either in an application specific fashion or, preferably, at the middleware layer.

Keywords: Energy-efficiency, mobile services, data transfer, application cooperation.

1 Introduction

The introduction of mobile services has resulted into increased need to transfer data between mobile phones and external computing devices, typically server computers. The communication between the mobile device and the server consumes energy. Extensive use of mobile services thus drains the battery and results into user dissatisfaction.

In this research, we investigate new possibilities for energy efficiency that are based on co-operation between different applications residing on the mobile device. In particular, we explore the effect of parallel voice calls and data transfers.

While some researchers have investigated the battery consumption of mobile applications [1-3], there is little work that studies the cooperation of different applications. Since the adoption of mobile services is still in an early state, it is not very common that a mobile phone accesses multiple services. However, this is likely to change when the popularity of mobile services grows. Intelligent cooperation between the different applications is likely to reduce needed energy in comparison to the case where each application in isolation tries to minimize its own energy consumption.

In spite of the emergence of new mobile services, using the mobile phone for conversations, voice calls, is likely to be an important use case also in the future. The idea we investigate in this paper is to delay non-time critical data transfers and perform them during voice calls. This kind of collaboration between the applications has potential for major energy savings. Once the radio of the mobile device is activated for voice communication, we can use it for the transfer of other data with little extra energy consumption. The bandwidth needed for voice is so small that in a typical network the radio channel can be used to transfer other data in parallel to the voice stream.

For many application the needed data transfer is not very time critical. For instance, downloading of upgrades, podcasts, emails, or RSS feeds as well as uploading of photos, backups, or calendar settings can be delayed in many cases without major effect on user experience. Naturally, the needs of different users and different applications vary but the set of services a user is using is likely to contain also a number of non-time critical ones. The possibility to delay some of the communication to happen in parallel with voice calls may allow us to develop new mechanisms for energy efficiency.

The key contribution of this research is to present the concept of delaying non-urgent data transfers so that they would take place during voice calls. We evaluate the proposed approach by power measurements with actual mobile phones. Finally we discuss the applicability of this approach and discuss the implementation alternatives.

The rest of the paper is structured as follows. Section 2 describes the situation when multiple services are in use in a mobile device. Section 3 presents our measurements that allow us to compare standalone data transfer with data transfer during a voice call in UMTS networks. Section 4 discusses the underlying mechanisms why data transfer in parallel with voice call is energy efficient. Section 5 discusses alternative architectural solution that would allow extending applications with this functionality. Section 6 discusses the solution and in particular evaluates its usage potential and its limitations. Section 7 reviews related research. Finally, section 8 concludes the paper and presents ideas for further research.

2 Interoperability of Mobile Service Data Transfer and Voice Calls

Currently, when multiple services are running on the same mobile device they typically schedule their activities independently without considering the other applications and services. Figure 1 shows an example of a case when the mobile phone is using email service and a photo upload service. The active periods of different applications are marked with colored rectangles.

Email service is an example of a periodic service. It checks periodically (with interval δ_{email}) if new mail has arrived. If new mail has arrived it is downloaded to the mobile phone. The length of the activity period varies as a function of the number and size of the email messages.

Photo upload is an example of a user (or application) triggered activity. User takes a photo and the system uploads it to a photo sharing service. The uploading activity is started by a user action (shooting the picture or decision to store it).

Finally, voice call activities are started at arbitrary time points either by the user or by calls coming in from the network.

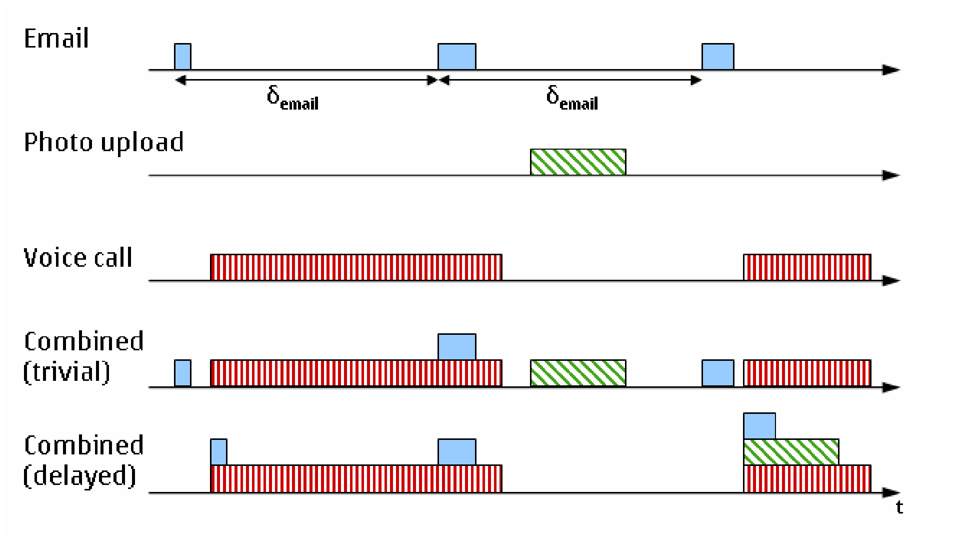

Fig. 1. Example sequence of events of three applications: email, photo upload, and voice call. Combined (trivial) shows the case where communication is started immediately for each application. Combined (delayed) shows the case where the communication is delayed to the time of the next voice call.

When we analyze the combined effect of these three services as depicted on the line “Combined (trivial)” in Figure 1, we can see that when the starting times of these activities are chosen independently from each other, they occupy a large part of the time axis. This means that whenever any single application is active and transferring data, the radio of the mobile phone has to be powered on. As a result the energy consumption of the combined use case is high.

However, if we have a possibility to delay the communication of the mobile services we can improve the situation. The line “Combined (delayed)” of Figure 1 shows the case when we delay the data transfer of non-urgent communications (email and photo upload activities in the example). Instead of following application specific schedules we put the data transfer needs into a waitlist and whenever a user makes or receives a voice call we activate the data transfers of the waiting services. As can be seen from Figure 1 the time when there are no ongoing data transfer activities is much longer when the delayed combination of different services is done.

Although the example shows the email and photo upload services as examples it should be easy to realize that there are a number of applications where such a combination makes sense. In fact, the more services the user is running on his device the higher the potential benefit from their smart combination.

Building on the idea of Figure 1 it seems promising to schedule the data transfer of mobile services so that as much of the data transfer activities are taking place at the same time. A very simple mechanism to achieve this is to delay the data transfer of non-urgent services until a time when the user is having a phone call. The exact benefit of this approach depends on the services the user is using, the available bandwidth, and on the pattern of events that take place. Naturally the potential savings also depend on how much extra energy is consumed when data transfer is happening in parallel to voice call and on the slowdown that simultaneous voice call causes for the data transfer. In the next section we investigate these aspects in detail.

3 Quantitative Evaluation with Nokia N95

In this section we evaluate the potential of the concept by analyzing the energy consumption of a mobile phone. We used Nokia N95 as the test device and performed the energy measurements with the Nokia Energy Profiler application that is available free of charge at Forum Nokia (www.forum.nokia.com). We performed the measurements in downtown office area in Helsinki, Finland, using the network of Elisa cellular operator.

Figure 2 shows the chart of an example measurement. The x-axis of the chart shows the time and the y-axis the power consumption. For this chart we downloaded a 2M email attachment with IMAP protocol using the native messaging application of the mobile phone.

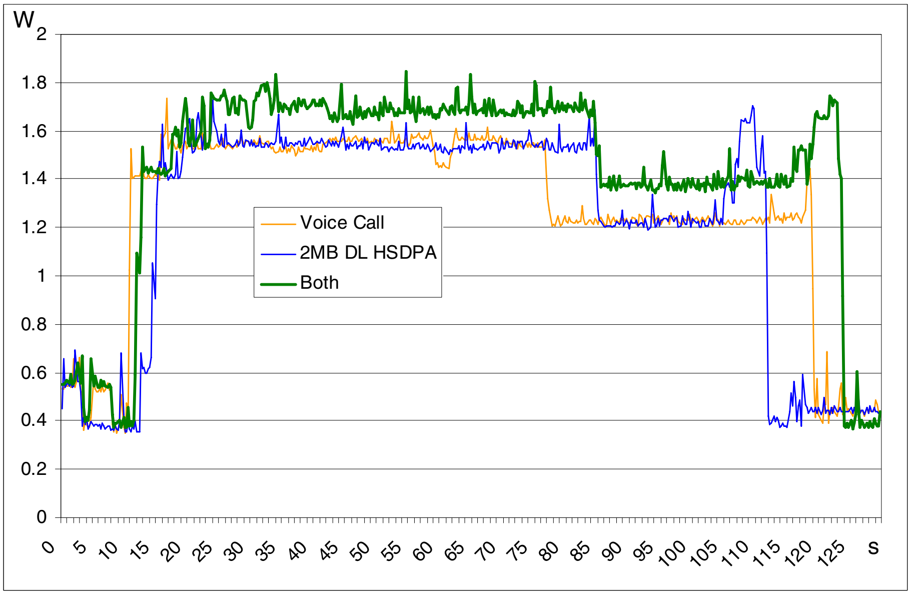

Fig. 2. Power consumption when a) a voice call is active, b) a data download is active, and c) both voice call and data download are active at the same time

The chart shows the energy consumption in three different cases:

- The light gray/orange curve shows the energy consumption when only a voice call is active. In the early part 15-80s the voice call is active and the display light is on. When the display light is switched off at 80s the energy consumption drops from 1.5 W to 1.2W.

- The dark grey/blue curve shows the energy consumption when the phone is downloading content. As the figure shows the energy consumption is rather similar than in the case of voice call. When the display light is on the power consumption is around 1.5W (20-90s and 110-115s). When the display light is off the power consumption is around 1.2W
- The thick gray/green curve shows a case when both voice call and data download are active simultaneously. The energy consumption is around 1.7W and 1.4 W when the display light is on and off respectively.

We can see from the combined curve than when data download is performed during a voice call the power consumption is only slightly higher than what the voice call would anyhow require. In this measurement the extra power is only 10%. Likewise we can see that the data transfer time increases when done during voice call but the increase is not very big, only 12%. As a result if we do the data transfer during a voice call we are able to perform the same activity with only 23% of the energy that it would take without the voice call.

Table 1. Measurement results comparing parallel data transfer during voice call with stand-alone data transfer

	Power (W)		Transfer time (s)		Energy (J)		
	Avg	Diff	Total	Diff	Total	Additional	Ratio
IMAP download of an email attachment (2.0 MB)							
Data transfer only	1.25		101		126	119	
Data transfer during voice call	1.39	11.2%	109	7.9%	152	16	14%
HTTP download of a 3gp video (2.1 MB)							
Data transfer only	1.22		109		133	125	
Data transfer during voice call	1.35	10.7%	124	13.8%	167	14	11%
Video Center download (10.1 MB)							
Data transfer only	1.23		476		585	552	
Data transfer during voice call	1.38	12.2%	488	2.6%	676	68	12%

The detailed measurement results are listed in Table 1. We measured three different cases:

- download of a 2.0 MB email attachment with IMAP protocol with the native messaging client,
- download of a 2.1 MB video with HTTP protocol with the native browser ("Einstein the bird" from www.free-3gp-video.com),
- and download of a 10.1 MB video with the Video Center application ("E90" from "Nokia N-series" channel).

To eliminate the variance of service load and network congestion we performed the measurements simultaneously with two similar Nokia N95 phones. One phone had a voice call connection during the entire download. The other one was only downloading. After the first measurement we swapped the roles of the phones and performed the same measurement again. The results are averages of these two measurements.

The power and transfer time columns show the measured values and their growth percentage when data transfer takes place in parallel with the voice call. The energy column shows the absolute energy of the each case.

The "Additional" subcolumn of energy shows the extra energy needed to perform the operation in comparison to the energy that the phone would anyhow consume during the same time. For the baseline values we use the phone idle power consumption (0.07W) for data transfer only and voice call power consumption (1.24W) for data transfers during voice calls. Finally the "Ratio" subcolumn shows the ratio of the additional energies.

The energy column, and especially its "Additional" subcolumn, shows clearly that the additional energy needed to transfer data during a voice call is small in comparison to the additional energy needed to transfer the same data without the voice call. This means that it is possible to save close to 90% of the data communication energy if we can perform the communication during voice calls.

This is a remarkable saving and it can be achieved with only a small increase in transfer time (between 3% and 14%). Naturally the biggest impact for the user is not the actual transfer time but the delay before the data transfer is started. The potential applications and the data transfer potential of the mechanism are discussed in greater detail in the following sections.

4 Background of the Phenomenon

The phenomenon that the measurements of the previous chapter capture can be explained by looking at the way the cellular radio in modern mobile phones is working. In terms of energy consumption the radio interface is the most power hungry component of a mobile phone accounting to around 50% of energy consumption in a connectivity use case [4]. The radio resources of a mobile phone are managed by a set of radio resource control algorithms that in addition to battery consumption influence signal quality, interference, mobility management and response time. (see e.g. [5] for details WCDMA radio resource management). While the details vary between different technologies the essential mechanism is that there are at least two main states. Idle state when power consumption is low and there is no traffic. Or active state, in which case the amount of transferred data has minor effect on the power consumption.

When the user is having a voice call the radio of his mobile phone has to be used to transfer the voice stream both to and from the mobile phone. Transferring the voice stream requires frequent activity from the radio circuitry so that there is not a possibility to power it down into an idle state. However, the radio interface of a mobile phone is able to handle a much larger bandwidth than what is needed for the voice. Whether this bandwidth is used or not does not have a major effect on the energy consumption. Therefore for the most energy efficient activity it makes sense to use as much of the available bandwidth as possible if the radio in any case is active.

5 Implementation

To illustrate the implementation alternatives we will use Nokia Podcasting application (http://europe.nokia.com/A4577364) as an example. Podcasting client is a good example of an application that can benefit from the mechanism because downloading podcast episodes in most cases is not time critical. The idea is to automatically download the new podcast episodes to your phone and listen or watch them later when you have time. Currently, the podcast application for Nokia phones only supports periodic downloading (every 15/60 min, twice/once a day). This will automatically turn on the selected radio at a specific time and start downloading eventual new episodes. If these automated downloads could be synchronized with phone calls, it would make a difference in power consumption with minimal change in the user experience.

On a general level the implementation will consist of the following steps:

- The application will register to a call listener, so that it can activate every time a call will be set up and deactivate when the call ends.
- In the podcasting case, when a phone call begins, the application would refresh the RSS feed and start/continue downloading the episodes when activated.
- When the phone call is terminated, podcasting application would be notified and it would pause ongoing downloads allowing radio to enter low-power state.

The same kind of functionality could be implemented to any application that does not require instant data transfer. For most applications this feature should be configurable in the options settings. For instance, busy executives may want to receive their emails immediately while private users may not mind if there is some delay before they receive their emails.

5.1 Application Level Implementation

There are at least two basic alternatives to implement the mechanism. It can be implemented at the application level or at the middleware level.

The straightforward approach is to implement the concept at the application level. Each application would create a listener to recognize when voice calls are started and finished. Managing the communication would then be completely application specific. This would be an easy extension to applications which already have a mechanism to manage the frequency of the communication. For instance, the Video center application has settings for feed subscriptions for video podcasts that allow selection of download times like night, morning, day, evening. It would be possible to add the new mechanism simply as another selection and implement the necessary functionality in the code.

Although the application level implementation is easy it has a number of limitations. First, each application would need to implement the same mechanisms. Different implementations of the same mechanism would increase the risk of mistakes and enlarge the code base. Providing the common mechanisms in some form of library would reduce this problem. Secondly, smarter interoperability between different applications would not be possible. For instance, there would be no way to manage the order of the different communication activities when a voice call starts. In some cases sequencing the communication activities may be a better alternative than having all of them run in parallel.

It would also be impossible to handle application specific deadlines in a cooperative way. We assume that many applications would need to limit the time how long the communication can be delayed. For instance, email user may require that the emails are synchronized once in every hour also in the absence of voice calls. If such deadline events are triggered independently by each application there would be no possibility to synchronize them. Instead, only the voice call initiation would allow the applications to synchronize their communication activities.

5.2 Middleware Level Implementation

The second way to implement the mechanism would be at the middleware level. In this case the basic mechanisms would be handled by the middleware and the applications would only perform the data communication according to the needs of the individual applications. The key parts of the mechanism would be made available to developers via an API. Note that in addition to the middleware support we would still need modifications to the applications using this mechanism. Applications would need to specify their communication policies and handle the details of the data transfer. These aspects are so application-specific that it is difficult to think how to do this without changes in the actual applications.

Figure 3 illustrates the middleware level implementation. The new component is the Schedule manager which is responsible for detecting the call event initiations and terminations, and signaling these to the applications.

When the Schedule manager is used the following sequence of steps takes place:

1. Applications will register to Schedule manager with a deadline value (max. delay for the download to start)
2. Incoming/outgoing call activates Schedule manager. Alternatively Schedule manager is activated when a deadline is reached.
3. Schedule manager gives a notification to applications to start data transfer
4. Applications start data transfer
5. When the voice call is terminated the Schedule manager gets a notification from the operating system.
6. Schedule manager tells the applications to finish their data transfer. Applications can react to this in an application specific way. For instance, they could continue the data transfer to a feasible termination point.
7. Applications can register a new deadline to schedule manager

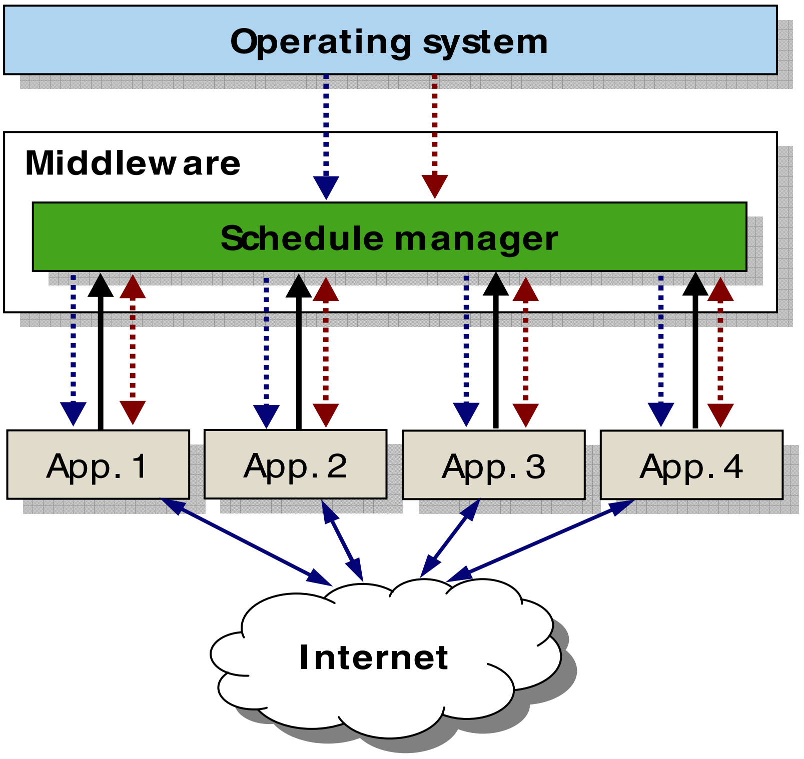

Fig. 3. Implementation of the concept as a schedule manager in the middleware level

The above sequence of steps allows room for extensions and added intelligence. Step 3 can be implemented in multiple ways. In the simplest case all registered applications would get the signal at the same time. In more sophisticated solutions the Schedule manager could further schedule the individual applications. For instance, for non-urgent bandwidth hungry applications a round robin scheduling might be better than activating all of them at a same time. This would need further mechanisms for the application to signal the Schedule manager when their data transfer activity has been finished.

Notice that even in the absence of voice calls the above mechanism would provide energy savings. The system would queue all delayed communication requests and once the deadline of the first request is reached the system could active also the other ones. The resulting coordinated transfer of data from different applications would reduce the need to power on and off the radio and thus decrease the energy consumption.

6 Discussion

6.1 Data Transfer Potential During Voice Calls

The potential for data transfer during voice calls is reasonably high as illustrated by the sample calculation in Table 2. According to StrategyAnalytics [6] the average

Table 2. Calculation showing the potential for data transfer during voice calls

Average call duration (min)	26
Average data transfer speed during call (kB/s)	30
"Low energy" data transfer potential (MB/day)	**46.8**
MP3 songs (#) [4MB/song]	12
video clip (min) [2.5 MB/min Reuters News]	19
email messages (#) [10kB/message]	4792

minutes of use (MOU) per subscriber per day is 26 min in US. With such talk times and assuming a relatively slow data transfer rate (30 kB/s) the daily potential for low-energy data transfer is already close to 50 MB. This amount of data is significant and allows the transfer of a large number of textual email messages and a considerable amount of multimedia content.

6.2 User Experience

In order to take advantage of this phenomenon for energy saving we need to postpone the data transfer to a later time point. For some applications such delay may not be tolerable. Fortunately, there is a large group of applications where the delayed data transfer is not likely to cause any major harm for the user experience.

Some of the candidate applications where such a feature is likely to be useful are:

- Synchronization: emails, calendar events, address book updates
- Downloading content to the phone: podcasts, video episodes, software upgrades, RSS feeds
- Uploading content to server: backups, photos

The essential concept does not depend on the kind of data that is being transferred or on the protocol used. The only essential aspect is that the user experience is not dependent on immediate action.

A further effect to the user experience is that notifications, for example of incoming mail, would arrive when the user is on the phone and displayed when the call ends. The potential benefit is less disruptions for the user, since the notifications from different services would come roughly at the same time and the user has already been interrupted by the voice call.

6.3 Limitations

Perhaps the biggest challenge for the proposed mechanism is to find a proper balance between user experience and energy efficiency. For some applications and some users the delay requirements are much stricter than for others. As discussed in the Implementation section the system would allow setting a maximum delay for each application. Naturally, the user could specify the delay but it would be yet another setting which would complicate the adoption of new service. How to provide the necessary flexibility but at the same time allow good enough control possibility to the user is a difficult question. The same question applies to many other parameters of mobile applications as well.

Our measurements show that the solution works nicely with current mobile phones. However, old model GSM phones supporting GPRS class B do not allow simultaneous voice and data connections. Therefore, even if the mechanism could be applied as a software upgrade also to older devices, it will not work unless the device supports class A functionality.

In a similar fashion it is unclear how the future communication technologies will benefit from the idea. Voice over IP packets would be competing of the same channel with the data transfer needs of the applications. Excessive use of data connections during VoIP call could influence the voice quality unless the packets are prioritized differently.

7 Related Research

The sensitivity of mobile devices to energy consumption has been widely recognized. Solutions to the energy consumption can be found on different levels. For instance, Keqiu et al. [7] is a rather recent survey of techniques for energy-efficient mobile computing analyzing the solution in different levels. They summarize research on methods and techniques at network interface, network protocols, operating systems, and application design layers. In addition to the energy-efficient communication the techniques for energy-efficient hardware and low-level software (see e.g. [8]) apply.

A lot of research has investigated the energy-efficiency of wireless sensor networks. Akyildiz et al. [9] present a good overview. More detailed, and more recent, survey of scheduling mechanisms in sensor networks is available by Wang and Xiao [10]. Anastasi et al. [11] is another survey of energy-efficiency in wireless sensor networks. Because the replacement of batteries is often very expensive in wireless sensor network the ability to maximize the run time of a wireless sensor is extremely important. One of the obvious solutions is to enter power saving mode where the activity of the wireless sensor is reduced for a period of time. In comparison to our research in sensor networks the traffic is often homogeneous. Often there is just a single application and in case of multiple applications there is seldom a dominant application. In mobile phones this is different since for most users the essential use is to make and receive phone calls.

A lot of the research on different communication layers has been layer specific and application independent. Obviously, generic application independent layered solutions are ideal in the respect that they do not limit the solution to a certain context. Anastasi et al. [12] is an example of the protocol level ideas to improve the energy-efficiency. They analyze the TCP protocol behavior for mobile web access. When fetching a web page consisting of multiple components they fetch components first to an intermediate access point so that device does not need to wait for the server processing. Thus device radio is on only during actual data transmission.

Many researchers see cross-layer design as a source of new solutions (see [13] for an overview). For instance, Anastasi et al. [14] show how taking the application level information into account can result into energy savings between 20% and 96%. The essence of their idea is that the mobile device is able to detect and differentiate between bursts and user think times. Our concept is similar but we apply it at higher protocol layers with different granularity. Instead of detecting burst and active communication intervals we detect voice calls and dynamically schedule activities for those time intervals.

In comparison to related research our approach is clearly application layer driven. Instead of focusing on the needs of a single application the starting point of our idea is the typical mix of different applications and services a user typically has (or is likely to have in the future). Furthermore it takes advantage of the fact that the dominant use case of mobile phone is voice calls.

The area of mobile services is so new that very few energy-efficient general mechanisms have been investigated. Most of the effort has been invested in novel innovative services. The energy-efficiency of the services often comes as an afterthought. Moreover, the battery consumption of each service is often optimized only within the context of the particular service. The fact that multiple services are likely to be used on the same device is frequently ignored.

There are a couple of studies which present ideas a bit analogous to ours but on different applications and domains.

Xiao et al. [15] propose an idea where the processing and user interface modules of the device are powered down when there is no incoming traffic. Our work shares the aspect with theirs that powering down components saves energy. In their work they always keep the communication interface powered up. However, in a modern mobile phone the communication is module is a major energy consumer and having a policy to allow it to sleep is very influential for the energy consumption of the device.

Conceptually a related system is in use at another abstraction layer in Linux operating system. The GLIB library used by GTK has a function, g_timemout_add_seconds, which will fire after a given number of seconds or when the first such timeout expires. GTK collects all timeouts to a single queue and all of them are executed at the same time when the earliest wake-up happens. This reduces the number of times the processor needs wake up and thus provides a drop in the power consumption [8].

A related concept is also available for energy-efficient disk I/O where the idea is to minimize the number of times the disk is spun up. The laptop mode of Linux kernel queues all write requests and executes them when the disk is spun up to serve a read request or when a timer value is exceeded [16].

8 Conclusions

The essential observation of our research is that there is a huge saving in battery consumption if we delay non-urgent data transfers and perform them while a voice call is active. Our measurements show that file transfer during voice call requires slightly over 10% extra power over the voice call and that simultaneous voice call slows down the file transfer only by 3%-14%. As a result we can save over 80% of energy in data transfer if we can delay the communication to a time when user is speaking at the mobile phone. For a typical user speaking 26 minutes a day there would be capacity for 50MB of data transfer during the voice calls.

It turns out that there is a large class of applications where the delayed data transfer would be applicable without major effect to the user experience. Examples of such application include synchronization (emails, calendar events, address book updates), downloading content to the phone (podcasts, video episodes, software upgrades, RSS feeds), and updating content to the server (backups, photos).

Since voice calls are the “necessary evil" for power consumption and require the radio to be active we should take advantage of these periods and transfer data at the same time.

The mechanism can be implemented either in an application specific fashion or, preferably, at the middleware layer. In any case applications need to be modified to take advantage of the mechanism.

With the increasing number and popularity of mobile services the idea has wide applicability. Enabling more energy-efficient use it can boost the adoption and use of mobile services and increase user satisfaction.

This paper presents and quantifies the basic phenomenon as well as discusses implementation alternatives. Further research would be needed on multiple areas. First, how sensitive are the users to the delayed content transfer? It seems that for many applications the delay is not an issue but this would require further confirmation. Second, what would be the optimal way to implement the solution and what kind API to use ssto expose the mechanism to application developers? Third, how to extend the idea to cover cases we have not investigated? Could the idea, for instance, work together with voice over IP? Finally, energy-efficient data transfer during voice calls could be a basic building block that might enable new ideas on mobile services and on their protocols.

References

1. Xiao, Y., Kalyanaraman, R.S., Ylä-Jääski, A.: Energy Consumption of Mobile YouTube: Quantitative Measurement and Analysis. In: Second International Conference and Exhibition on Next Generation Mobile Applications, Services and Technologies, Cardiff, Wales, UK (2008)
2. Nurminen, J.K., Nöyränen, J.: Energy-Consumption in Mobile Peer-to-Peer – Quantitative Results from File Sharing. In: 5th IEEE Consumer Communications & Networking Conference CCNC 2008, Las Vegas, Nevada (2008)
3. Kelenyi, I., Nurminen, J.K.: Energy Aspects of Peer Cooperation - Measurements with a Mobile DHT System. In: IEEE CoCoNet Workshop 2008 Cognitive and Cooperative Wireless Networks collocated with IEEE ICC 2008, Beijing, China (2008)
4. Neuvo, Y.: Cellular phones as embedded systems. In: IEEE International Solid-State Circuits Conference, Digest of Technical Papers, vol. 1, pp. 32–37 (2004)
5. Holma, H., Toskala, A.: WCDMA for UMTS. John Wiley & Sons, Chichester (2000)
6. Wireless Operator Performance Benchmarking Q2 2008, Strategy Analytics (2008)
7. Keqiu, L., Nanya, T., Wenyu, Q.: Energy Efficient Methods and Techniques for Mobile Computing. In: Third International Conference on Semantics, Knowledge and Grid, pp. 212–217 (2007)
8. Garrett, M.: Powering down. Commun. ACM 51(9), 42–46 (2008)
9. Akyildiz, I.F., Su, W., Sankarasubramaniam, Y., Cayirci, E.: Wireless sensor networks: a survey. Computer Networks 38(4), 393–422 (2002)
10. Wang, L., Xiao, Y.: A survey of energy-efficient scheduling mechanisms in sensor networks. Mob. Netw. Appl. 11(5), 723–740 (2006)
11. Anastasi, G., Conti, M., Di Francesco, M., Passarella, A.: Energy Conservation in Wireless Sensor Networks: a Survey. Ad Hoc Networks 7(3) (2009)

12. Anastasi, G., Conti, M., Gregori, E., Passarella, A.: Performance comparison of power-saving strategies for mobile web access. Perform. Eval. 53(3-4), 273–294 (2003)
13. Srivastava, V., Motani, M.: Cross-layer design: a survey and the road ahead. IEEE Communications Magazine 43(12), 112–119 (2005)
14. Anastasi, G., Conti, M., Gregori, E., Passarella, A.: 802.11 power-saving mode for mobile computing in Wi-Fi hotspots: limitations, enhancements and open issues. Wirel. Netw. 14(6), 745–768 (2008)
15. Xiao, Y., Chen, C.L.P., Kinateder, K.K.J.: An optimal power saving scheme for mobile handsets. In: Sixth IEEE Symposium on Computers and Communications, pp. 192–197 (2001)
16. Samwel, B.: Kernel korner: extending battery life with laptop mode. Linux J. 2004(125), 10 (2004)