Improving Smartphones Battery Life by Reducing Energy Waste of Background Applications

Bolla, R., Khan, R., Parra, X., & Repetto, M. (2014). Improving Smartphones Battery Life by Reducing Energy Waste of Background Applications. In 2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies: Proceedings. (pp. 123-130). (International Conference on Next Generation Mobile Apps, Services and Technologies). DOI: 10.1109/NGMAST.2014.10

Published in:
2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies: Proceedings

Improving Smartphones Battery Life by Reducing Energy Waste of Background Applications

Raffaele Bolla [*] , Rafiullah Khan [*] , Xavier Parra [†] and Matteo Repetto [‡]
[*] *DITEN Dept. University of Genoa, Genoa, Italy,* Email:{raffaele.bolla, rafiullah.khan}@unige.it
[†] *Universitat Politecnica de Catalunya, Vilanova i la Geltru, Spain,* Email: xavier.parra@upc.edu
[‡] *CNIT, Research Unit of University of Genoa, Genoa, Italy,* Email: matteo.repetto@cnit.it

***Abstract*—Smartphones have undergone a remarkable evolution over the last few years, from simple calling devices to full fledged computing devices where multiple services and applications run concurrently. Unfortunately, battery capacity increases at much slower pace, resulting as a main bottleneck for Internet connected smartphones. Several software-based techniques have been proposed in the literature for improving the battery life. Most common techniques include data compression, packet aggregation or batch scheduling, offloading partial computations to cloud, switching OFF interfaces (e.g., WiFi or 3G/4G) periodically for short intervals etc. However, there has been no focus on eliminating the energy waste of background applications that extensively utilize smartphone resources such as CPU, memory, GPS, WiFi, 3G/4G data connection etc.**

In this paper, we propose an Application State Proxy (ASP) that suppresses/stops the applications on smartphones and maintains their presence on any other network device. The applications are resumed/restarted on smartphones only in case of any event, such as a new message arrival. In this paper, we present the key requirements for the ASP service and different possible architectural designs. In short, the ASP concept can significantly improve the battery life of smartphones, by reducing to maximum extent the usage of its resources due to background applications.

***Index Terms*—Green networking, smartphone, energy efficiency, battery life, application state proxy.**

I. Introduction

Today, smartphones are equipped with high memory and processing capabilities. They offer high speed Internet connectivity through EDGE, 3G/4G or WiFi interfaces and embed different type of sensors, including GPS, compass, gyroscope, proximity and health related sensors [1]. Due to enormous popularity with over 2 billion devices in use, the number of applications exploded (especially on Android and Apple stores) over the last few years. Most of the interesting applications such as Voice-over IP (VoIP) and Instant Messenging clients: Viber, Vonage, Whatsapp, Skype, Facebook messenger etc run in background and periodically transmit/receive status messages; they not only utilize resources such as CPU and memory, but also demand full time Internet connectivity, thus resulting in much reduced battery life. Previous studies revealed that the Internet connectivity constitutes about 62% of power consumption for a mobile device in idle state (with LCD and backlight in OFF state) [2]. Further, 3G/HSDPA data connectivity is more power hungry compared to WiFi (as shown in Fig. 1) especially when the small size packets are more frequently exchanged [3].

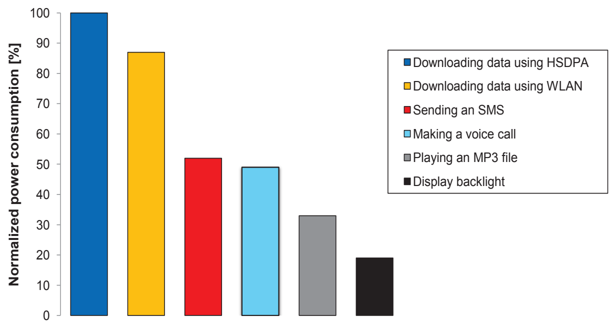


Figure 1. Power consumption of different types of services normalized to the HSDPA on Nokia N95 [3].

Several strategies have been proposed in the literature for improving the battery life of smartphones. Most of them focuses on compressing data or shaping network traffic to create short idle periods during which network interfaces can stay in the low power states [1]. However, no strategy has been proposed till now to eliminate the energy waste due to background applications which periodically transmit/receive presence or heartbeat messages. Although, Internet interfaces usually have different operational states; the frequent heartbeat messages from many background applications force them to stay always in active state. It is important to note that frequent states switching between high power active and low power idle incur significant additional energy overhead [4]. Fig. 2 shows the distribution of consecutive packets inter-arrival times on a typical smartphone [1]. About 40% of the packets have 0.5 ms inter-arrival time or even less. The energy consumed by Internet interfaces mainly depend on the size and frequency of heartbeat messages transmitted/received by applications. Thus, an Internet connected smartphone normally consumes battery 3 to 4 times faster depending on the number of application running in the background.

In this paper, we present an Application State Proxy (ASP) that suppresses/stops the background Internet-based applications on smartphones and maintains their presence on any other network device. This way, the energy consumption significantly decreases, due to reduced CPU and memory utilization and network interfaces (such as WiFi or 3G/4G) mostly stay in low power states. The ASP concept is different from the commonly adopted strategy of partial offloading of heavy computational tasks to the cloud (thin client-server

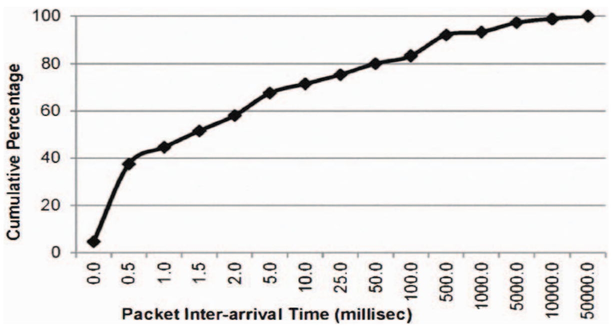


Figure 2. Analysis of packets inter-arrival time on a typical smartphone [1].

scenario). The ASP offloads complete applications to another network device to extensively reduce the load on smartphone resources. The ASP concept is basically the extension of our previous work on the Network Connectivity Proxy (NCP) for desktop computers [5], [6]. The NCP maintains presence of desktop computers during their sleeping periods. Unlike the NCP, the ASP only needs to maintain the applications presence on behalf of smartphones. The ASP maintains the applications presence as long as there are no events, such as new message arrival. When any event occurs, the ASP immediately returns/resumes the application back on the smartphone to timely inform the user. In this work-in-progress paper, we have mainly addressed the key requirements for the ASP service and proposed different possible architectural designs.

The rest of our paper is organized as follows. Section II briefly addresses previous work in literature. Section III presents the ASP concept and addresses basic requirements. Section IV describes the possible architectural designs. Section V presents different techniques for proxying the presence of applications. Section VI describes the design of our communication protocol. Section VII addresses the expected benefits. Finally, Section VIII concludes the paper.

II. Related Work

A brief survey on the energy consuming entities on the smartphone platform has been presented in [7]. The authors have taken into consideration the components including CPU, display, memory, mp3 player, wireless interfaces (e.g., bluetooth, WiFi, 2G, 3G) etc to give an understanding of the most power-hungry parts in a smartphone. One of the most accurate method to measure the energy consumption on Android platform by a specific application has been presented in [8]. The proposed approach is based on the kernel activity monitoring to determine the accurate usage of different hardware components by an application. A detailed analysis of advances in the battery capacities and understanding of more energy-hungry aspects of an application have been presented in [9], [10]. The authors have pointed out that the Internet-based applications usually consume a major share of the smartphone battery life.

The random data transmission by recurrent applications significantly limits the smartphone battery life. M. Calder et al. in [11] presented a batch scheduling mechanism for smartphones. The authors have quantified the possible energy savings through batch scheduling on real mobile platforms (e.g., Android) with particular focus on maximizing phone's sleep time or minimizing the frequency of wake-up by recurrent applications. Some further work in this domain for delay tolerant applications has been addressed in [1], [12] and [13].

Data offloading from 3G/4G network to WiFi is another strategy for improving the smartphone's battery life. From the perspective of energy required for data transfer, the WiFi is much more efficient than 3G/4G. This domain has been explored by N. Ristanovic et al. in [14] and K. Lee et al. in [15]. A useful strategy for improving the smartphones battery life is the computational load offloading to the cloud (thin client-server scenario). However, this strategy was basically proposed to execute complex applications on any smartphone even with low computational capabilities. Microsoft office 365 and Matlab for portable devices are based on this strategy. Some interesting work in this domain is published in [16] and [17].

Until now, no strategy has been proposed for smartphones to eliminate the energy waste due to applications running in the background which not only utilize CPU and memory but also extensively use the Internet interfaces. In this paper, we propose the ASP concept to achieve this objective. The ASP is the continuation of our previous work on NCP, which impersonates the presence of desktop computers during their sleeping periods [5], [6]. Unlike the NCP, the ASP maintains only the applications presence as the smartphones are always powered-up devices.

III. Application State Proxying

Many Internet-based applications on smartphone run in the background to send/receive periodic presence or heartbeat messages. These periodic messages keep the application state up-to-date and inform the user on time in case of any event. Further, these messages also update the remote peers about the user presence. The smartphone Internet interfaces (such as WiFi and 3G/4G) mostly stay in active mode due to frequent heartbeat messages generated by many background applications (depending on the number of applications installed on smartphone and their heartbeat message periods). The best option to reduce energy waste is to reduce the usage of smartphone's resources by the background applications. This objective can be achieved if the smartphone suppresses/stops the background applications from utilizing its resources while still maintaining their presence. A proxy can be the optimal solution to achieve this objective.

A. *Overview*

We propose the ASP as the optimal solution to improve the smartphone battery life. The ASP is a software entity running on any device in the home network (e.g., Home Gateway (HG)) and maintains the presence for smartphone applications. The ASP resumes the specific application back on smartphone only when any event is received e.g., a new message. Thus, the applications run most of the time on ASP hosting device

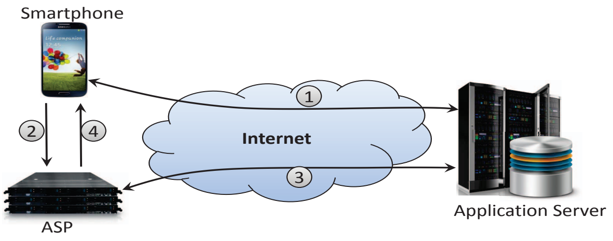


Figure 3. The ASP functional view: (1) Application state maintained by the smartphone, (2) Smartphone requests ASP to impersonate its application presence, (3) The ASP impersonate the application presence, (4) The ASP receives application event and transfers the application presence back to smartphone.

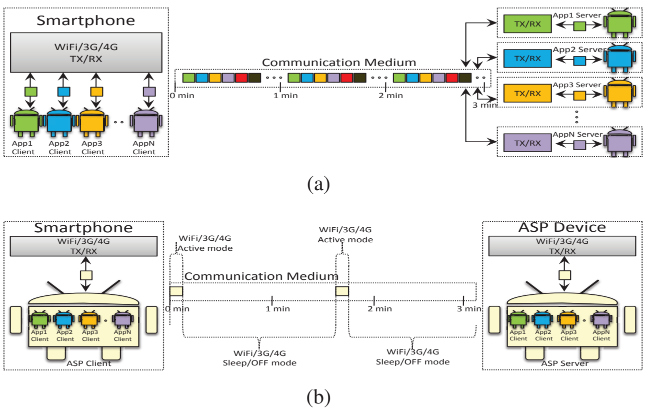


Figure 4. The ASP concept: (a) Normal scenario of background running applications, (b) The ASP scenario by efficiently controlling background applications (communication medium between the ASP server and each application server is not shown but will be similarly congested as in (a)).

and run on smartphone for very short periods (only when any event is received). Since the smartphone is a mobile device, it should be able to request ASP service both from inside and outside the home network. However, to access the ASP service from outside the home network will require addressing the NAT and Firewall issues. The optimal location for ASP service is the HG which is a low power device and always remains powered-up, thus resulting in very low incremental network energy consumption. The benefits that can be achieved from the ASP service include: *(i)* longer battery life due to much reduced utilization of smartphone's hardware components such as CPU, memory, WiFi, 3G, 4G etc, and *(ii)* lower utilization of 3G/4G data which is usually limited by service providers.

The generic functional diagram for ASP is shown in Fig. 3. It consists of four main steps:

1) The application is active on the smartphone and directly communicates with the application service provider. This can be a short interval during which the user answers a call or reads a new received message.
2) The user requests the ASP to maintain its application presence. Meanwhile, it stops the smartphone application to reduce utilization of hardware resources such as CPU, WiFi, 3G/4G etc. This step can be based on the user command or inactivity timer expiry for the application.
3) Now the ASP maintains the application presence directly with the application service provider on behalf of the smartphone. The ASP impersonates the application presence as long as no event is received.
4) The ASP resumes the application back on smartphone when any event is received. Now all steps in Fig. 3 repeat over and over again.

Fig. 4 gives a more clear understanding of the ASP concept. It can be observed (in Fig. 4(a)) that the smartphone is running concurrently many different applications in the background which periodically exchange heartbeat messages with their respective application server. Thus, the WiFi or 3G/4G mostly stays in the active state to transmit very large number of packets on the communication medium. The Internet interfaces may also have frequent operational state switching which results in significant additional energy overhead. Fig. 4(b) depicts the scenario where smartphone is taking benefit from the ASP service. It can be observed that very few packets are exchanged over the communication medium between the smartphone and ASP hosting device. These few packets may correspond to infrequent ASP client advertisements to ensure the smartphone availability to the ASP server or may indicate effective communication between the ASP client and server to stop or resume a specific application on either side. Now the ASP server manages these background applications and periodically exchange their heartbeat messages with their respective application server. It is quite obvious in Fig. 4(b) that the smartphone can find long idle periods during which its hardware components (e.g., WiFi or 3G/4G) may switch to low power states or the smartphone operates at lower CPU rate due to less processing requirements when no background applications are running. Generally speaking, Fig. 4(b) gives the idea of using a common server which is responsible for managing heartbeat messages for all applications.

B. Basic Requirements

We divided the requirements into three categories: *(i)* smartphone requirements, *(ii)* ASP requirements and *(iii)* communication requirements.

1) Requirements for the Smartphone: The following are some of the basic requirements for the smartphone:

1) Have the capability to stop/resume the applications. This can be based on inactivity timer expiry or based on the request of user or ASP service.
2) Be able to resume an application in smallest possible time. This is especially true for applications requiring immediate user response/attention in a limited time span e.g., a call on Viber. However, the delay will not cause any adverse effects for events such as a new message arrival Facebook messenger, Viber, Whatsapp, Vonage or updates for Dropbox, Box etc.
3) Have the ability to scale CPU performance based on required level of processing and/or putting the hardware components such as WiFi, 3G/4G etc into low power idle state. This step is critically important to significantly extend the smartphone battery life.

4) Ability to detect and prevent an application from stopping if it is actively performing some online activity e.g., a user chatting with his friend.
5) Should have unique identity and should be accessible by ASP service at any time.

2) Requirements for the ASP Service: The following are some of the basic requirements for the ASP service:

1) Should be always available. It is necessary to run the ASP service on a device that is always powered ON and connected to the Internet e.g., HG.
2) Should have the ability to maintain presence for different applications on behalf of smartphone. This includes generating/responding to routine application specific periodic heartbeat messages.
3) Should have the ability to detect the events for different application and timely resume the applications back on smartphone.
4) Should provide proper user privacy and data security. Isolation of data from different users is quite important.
5) Should be able to provide services to more than one user or smartphone. The scalability is important as the home network may have many smartphones.
6) Should have unique identity and should be easily accessible by the smartphone at any time.

3) Communication Requirements: We have proposed a cooperative approach that allow the smartphone and ASP to communicate with one another. The following are some of the basic requirements for communication between the smartphone and the ASP service:

1) The communication protocol should have proper security measures. This is especially important when the smartphone is outside the home network and the communication passes through public infrastructure.
2) The communication link should be established in shortest possible time and should have low latency and overhead.
3) Autonomous seamless communication requiring no or minimal configurations can be the optimal choice. This requirement can be easily met if both the smartphone and ASP lie in the home network, however special techniques or strategies need to be adopted for communication over Internet.
4) Both the smartphone and the ASP needs to have unique identity to be able to communicate. This requirement can be easily met if both the smartphone and ASP lie in the home network, however NAT and Firewall issues need to be addressed if the smartphone is located outside the home network.

IV. Possible Architectural Designs

The ASP for smartphones can have three possible designs based on its location.

A. Local Deployment of ASP

The generic design for the Local ASP (L-ASP) deployment is shown in Fig. 5. The ASP and smartphone both are located

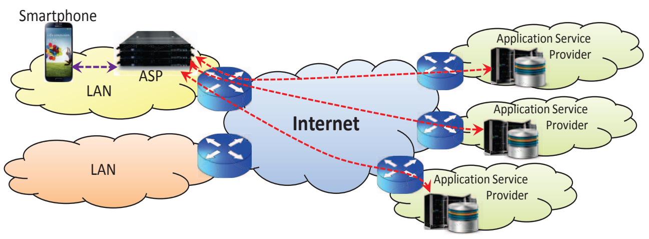

Smartphone
ASP
LAN
Internet
LAN
Application Service Provider
Application Service Provider
Application Service Provider

Figure 5. The L-ASP architecture: The purple line depicts the transfer of applications control between the smartphone and the ASP. The red lines depict the maintenance of different applications presence by ASP on-behalf of smartphone.

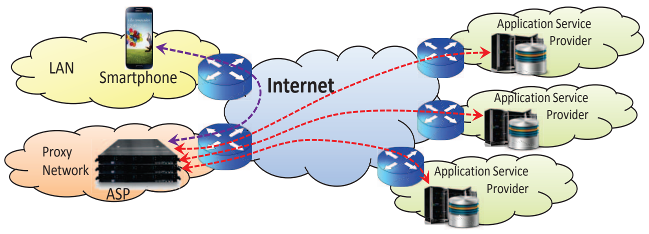
in the same network. This is the typical scenario when the user is at home and his smartphone is connected with the ASP service provided by a local device. The ASP service may be provided by the switch/router, HG or a standalone device such as desktop computer, laptop, tablet etc. Switch/Router can be the optimal place for the ASP service as it is a low power entity and always stays powered ON and connected to the Internet. A standalone device will provide higher memory and processing capabilities but will result in the network energy wastage if it is kept powered-up 24/7 just for the purpose of ASP service. The L-ASP deployment also offers the advantage of easy communication between the smartphone and ASP. There are many techniques that can be adopted for zero-configuration, auto discovery/communication and seamless networking between the smartphone and ASP without any need of user configurations. Two most common techniques are multicast DNS (mDNS) (mostly used by Apple devices) and Universal Plug & Play (UPnP) (fast emerging standard for network devices).

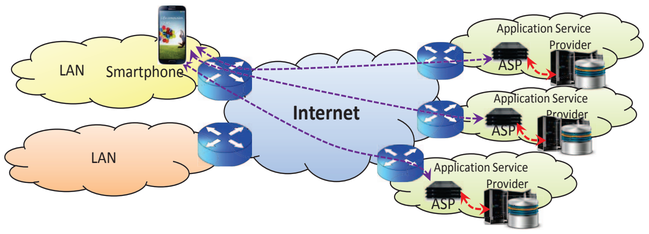
Besides the benefits offered by the L-ASP deployment, there are two main issues:

1) The smartphones are usually at home for very less time e.g., almost 10 hours at night or outside work hours. Since the main purpose of smartphone is mobility and portability, it will not be able to benefit much from the ASP service unless the L-ASP deployment is available everywhere (also at work places).
2) The switch or router usually have limited memory and processing capabilities which makes difficult for the ASP service to implement proxying for large number of complex applications.

B. General Internet-wide ASP

The design of General Internet-wide ASP (GI-ASP) is shown in Fig. 6. It can be observed that the ASP and smartphone are located in the different networks. The GI-ASP deployment is the typical scenario when the smartphone is benefiting from the ASP service located anywhere in the world. The ASP service will be provided by a powerful device that is highly scalable and capable to proxy large number of complex applications. The power consumption of the ASP hosting device is not a concern as it will spread over very large number of smartphones resulting in negligible incremental


LAN
Smartphone
Internet
Proxy Network
ASP
Application Service Provider
Application Service Provider
Application Service Provider
Figure 6. The GI-ASP architecture: The purple line depicts the transfer of applications control between the smartphone and the ASP. The red lines depict the maintenance of different applications presence by ASP on-behalf of smartphone.


LAN
Smartphone
Internet
LAN
Application Service Provider
ASP
Application Service Provider
ASP
Application Service Provider
ASP
Figure 7. The ASI-ASP architecture: The purple line depicts the transfer of applications control between the smartphone and the ASP. The red lines depict the maintenance of different applications presence by ASP on-behalf of smartphone.

power per smartphone. Further, the ASP service will be always accessible from everywhere and smartphones can benefit from it 24/7.

Besides many benefits that can be achieved by the GI-ASP deployment, there are also some issues and challenges. The communication between the smartphone and ASP will be quite challenging as the smartphones are usually behind NAT and Firewall (preventing ASP to connect with smartphone in case of an application event). Also, there will be no autonomous discovery or seamless communication and a minimal user configurations may be required.

C. *Application Specific Internet-wide ASP*

The design of Application Specific Internet-wide ASP (ASI-ASP) is shown in Fig. 7. It can be observed that each application service provider also offers the ASP service to its clients. In other words, each ASP offers proxying for one specific application. The smartphone can request each ASP separately to maintain specific application presence on its behalf. The ASI-ASP deployment will not face any privacy concerns as the proxying service is provided by the actual application developers. Thus, this deployment is quite suitable for closed-source proprietary applications. Instead of using the ASP, the application service provider may include application freeze functionality that will contact the application client on smartphones only when there is any event.

The ASI-ASP scenario is quite simple and has no complex requirements. The application developers need to support proxying for their clients and include an extra feature in the client applications on smartphones to stop or resume the proxying service. When the user wants to save energy on his smartphone, he will request the application end-point for proxying. The application service provider will apparently disconnect the connection with smartphone but preserve its presence. Based on user request or any application event, the application service provider will return the application control back to smartphone. Further, the ASI-ASP scenario is suitable because the smartphone can access it from anywhere. Security, privacy, flexibility and scalability will be the concerns for application developers in general but not specific to the ASP service.

V. Applications Proxying Techniques

The main objective of the ASP is to proxy applications until an event is received. Application events can be classified into two main categories: *(i)* delay sensitive and *(ii)* delay insensitive. Delay sensitive events (such as an incoming call) require user response within the shortest time span. Thus, the ASP needs to resume the application back on smartphone in the shortest possible time to avoid any adverse effects. On the other hand, the ASP may take few seconds to resume the application back on smartphone in case of delay insensitive events (such as new message arrival). Most of the events related to smartphone applications are delay insensitive such as messages on Facebook messenger, Vonage, Viber, Whatsapp, Twitter, Skype or updates on Dropbox, Box, etc. Resuming an application on smartphone for delay insensitive events is less challenging. Simply starting up the specific application on smartphone will automatically receive the events from the application service provider. Special care is required when returning the application control for delay sensitive events. Usually, an application (with same user account) running on different devices receive events simultaneously. E.g., a Skype or Viber client running on two different devices will receive the call event simultaneously. Thus, just starting up the specific application on smartphone will automatically receive event (a call in this case) if the delay is not significant.

The ASP impersonates presence of applications on behalf of smartphones by sending/receiving periodic application specific heartbeat messages. The ASP can achieve this objective by one of the following techniques.

A. *Using Application Specific Stubs*

The application specific stub is a piece of software derived from the actual application source code that periodically generate or respond to the heartbeat messages. Writing the application stub requires application source code and understanding of its functionalities. The stub only contains the subset of application functionalities which are enough to impersonate the application presence. It is usually not feasible for the stubs to include the entire application source code as the application may also depend on the hardware components or drivers e.g., display, memory, disk etc that may not be available or may overload the ASP hosting device. Another important part of the stub is to understand or recognize events which will require to resume application back on smartphone. The application stubs usually also require login credentials of the application account. The idea of stubs was initially introduced by Y. Agarwal et al. in [18]. The authors have developed a USB based architecture ‘Somniloquy’, that maintains the presence of sleeping desktop computers. Since the stubs require application source code, this approach can not be easily adopted for the proprietary closed source applications.

B. Using Virtual Machines

The ASP may also use virtual machines to impersonate the presence of smartphone applications. For each smartphone, the ASP will instantiate the corresponding image. The image is responsible to maintain the presence of all applications running on that specific smartphone. Since, the virtual machine runs a complete operating system, it is easy to proxy many open source and proprietary closed source applications. The ASP will simply start the specific applications on virtual machine when the smartphone requests its service. However, some strategies need to be adopted to detect the events which require the application control transfered back to smartphone. Here the transfer of application control may simply imply stopping or running the specific application on that device. E.g., when a new message arrives, the ASP only needs to inform the smartphone to run the specific application that has new event. The application will update itself automatically after start up. This can be also true for delay sensitive events e.g., a call if the delay is not significant. One main limitation of this approach is the resources required to run a virtual machine. This approach can be good if the ASP runs in home network and manages only few smartphones. However, scalability will be a serious concern if the ASP runs as one global entity managing thousands of smartphones.

C. Using Generalized Heartbeat Messages

The ASP may use a generalized heartbeat message template that is suitable to generate heartbeat messages for any application. This template needs to be filled up by the applications running on the smartphone. The template will contain all the required fields and strategies to generate the periodic heartbeat messages. Thus, the applications on smartphone need to implement two features: freeze and resume. The application freeze feature fills up the heartbeat message template at the ASP. After that the application on smartphone will stop generating or sending heartbeat messages, instead the ASP will start sending periodic heartbeat messages based on the information provided in the template. When the ASP receives an application event, it will invoke the application resume feature on the smartphone. After the application resume, the smartphone will once again start sending/responding to heartbeat messages.

It can be observed that this approach is future oriented and requires the applications to implement freeze and resume features. Indeed, this approach is quite suitable for the ASI-ASP scenario depicted in Fig. 7 where each application service provider also provides the ASP service to its clients.

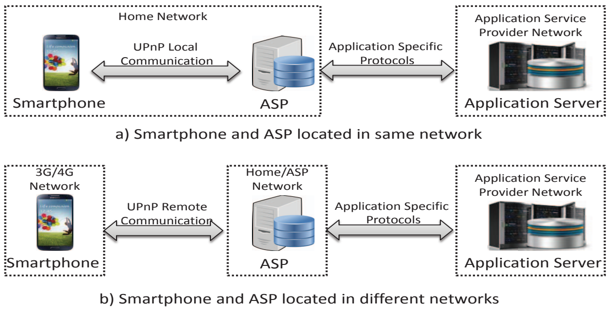

Home Network
UPnP Local Communication
Application Specific Protocols
Application Service Provider Network
Smartphone
ASP
Application Server
a) Smartphone and ASP located in same network
3G/4G Network
Home/ASP Network
UPnP Remote Communication
Application Specific Protocols
Application Service Provider Network
Smartphone
ASP
Application Server
b) Smartphone and ASP located in different networks

Figure 8. Possible scenarios for ASP service: (a) Smartphone and ASP are located in same network and communicate using standard UPnP architecture. (b) Smartphone and ASP are located in different networks and communicate according to URA specification.

VI. Communication Protocol

The communication protocol is mainly required for transfer of applications control between the smartphone and ASP. Different protocols can be considered for the ASP framework which can lead to an efficient, flexible and scalable solution. We have designed our communication framework using UPnP architecture [19]. Our choice for UPnP architecture is well motivated due to its interesting features of zero configuration, auto-discovery and seamless communication ease. Further, the UPnP protocol gained enormous popularity for future network devices and can be easily supported by heterogeneous network devices including smartphones, printers, scanners, copiers, Internet gateways etc. The UPnP architecture is built upon several different protocols: *(i)* Simple Service Discovery Protocol (SSDP) to search/advertise device presence, *(ii)* General Event Notification Architecture (GENA) protocol for notification about device events and *(iii)* Simple Object Access Protocol (SOAP) which is used to send actions/commands to a UPnP device. The SOAP and GENA protocols usually operate on the top of HTTP protocol while SSDP uses HTTPU (an extension of the HTTP protocol which uses UDP as the transport protocol instead of TCP). The UPnP Device Architecture (UDA) specified two main roles for the UPnP devices: Controlled Device (CD) and Control Point (CP). The CD represents the physical entity that implements one or more services. Each service consists of one or more actions that build remote procedure calls. The CP runs on another network device that sends commands to specific service of the CD and invokes particular action.

The UPnP protocol was originally proposed for seamless autonomous communication between devices in local network. However, the smartphone is a mobile device and mostly stays outside the local/home network. Thus, we have tailored our design to support both, the communication in local network as well as over the Internet. The two possible scenarios are depicted in Fig. 8. Fig. 8(a) depicts the scenario when the smartphone is present at home and connected to the home gateway which is offering the ASP service. It is quite important in this scenario that the smartphone is not using 3G/4G data network, instead using the Internet over home WiFi.

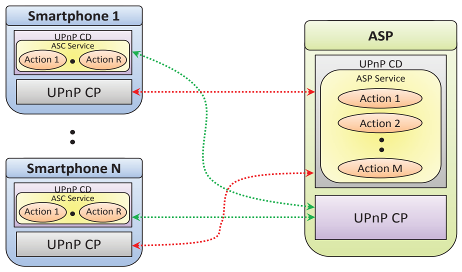


Figure 9. UPnP model for local communication framework.

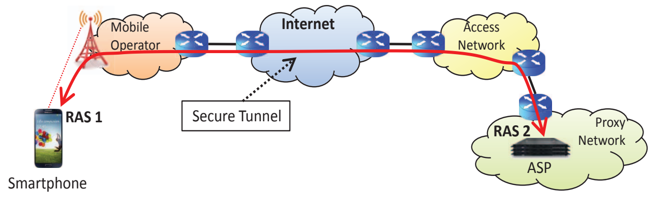


Figure 10. UPnP Remote Access scenario.

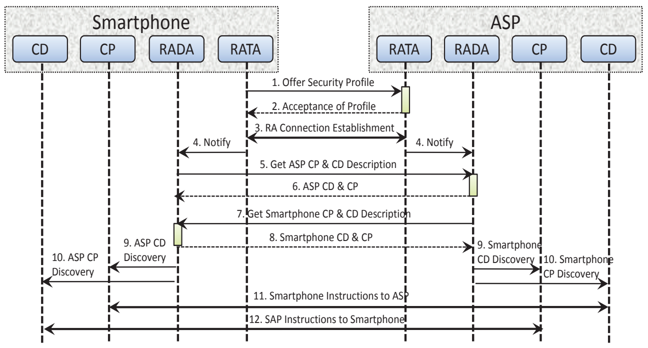

Smartphone
CD
CP
RADA
RATA
ASP
RATA
RADA
CP
CD
1. Offer Security Profile
2. Acceptance of Profile
3. RA Connection Establishment
4. Notify
4. Notify
5. Get ASP CP & CD Description
6. ASP CD & CP
7. Get Smartphone CP & CD Description
8. Smartphone CD & CP
9. ASP CD Discovery
10. ASP CP Discovery
9. Smartphone CD Discovery
10. Smartphone CP Discovery
11. Smartphone Instructions to ASP
12. SAP Instructions to Smartphone

Figure 11. Simplified communication scenario between smartphone and ASP.

Fortunately, due to comparatively less power consumption by WiFi than 3G/4G, the modern smartphones automatically switch to WiFi whenever an active access point is discovered. Fig. 8(b) represents the scenario when the smartphone is out of the home network and tries to avail the ASP service offered by a device at home or a device located anywhere in the world. Since the smartphone and ASP hosting device are located in two different networks, the communication between them needs to address many issues and challenges to ensure reliability and security. The UPnP Remote Access (URA) specification has recently been proposed that allows secure communication between two device located in different networks in a similar way as if they are located in the same network [20].

A. *Local communication framework*

The generic design of our UPnP based local communication framework is depicted in Fig. 9. In the general UPnP communication scenario, each physical device either implements a CP or a CD. However, in our design in Fig. 9, both the smartphone and the ASP implement a CP as well as a CD. The CP is required on smartphone to send proxying requests for different applications to the service offered by the CD of ASP. Similarly, the CP is required on the ASP to send commands for resuming applications back on smartphone (in case of any event). Thus, the Application State Control (ASC) service on smartphone provides different actions with remote procedure calls for stopping and resuming different applications.

B. *Internet communication framework*

The Internet communication framework is more complex as the ASP and smartphone are located in different networks. The URA specification can be adopted that extends the UPnP coverage beyond LAN boundaries and enables the CP/CD located in one network to securely communicate in a seamless way with another UPnP CD/CP located in a remote network. The URA specification also addresses key challenges and issues arise during communication between devices located in two remote networks (e.g., NAT and Firewall issues, quality of service etc) [20].

The URA specification proposed usage of Remote Access Server (RAS) running on each device. The RAS consists of Remote Access Discovery Agent (RADA) and Remote Access Transport Agent (RATA). The RADA exposes the devices and services available in the local network to another RADA in the remote network. The RATA is responsible to establish a secure connection between two remote RAS. The typical scenario is depicted in Fig. 10. The remote access connection can be initiated from either side, the smartphone RAS or the ASP RAS. The smartphone RAS exposes its embedded UPnP CD and CP to the RAS on ASP. Similarly, the RAS on ASP hosting device exposes its embedded or local UPnP CD and CP to the smartphone. For successful establishment of connection, the smartphone RAS needs to know the remote ASP RAS IP address and other security associations and vice versa. This information is stored in a Remote Access Application Server (RAAS) managed by the service provider. The smartphone retrieves this information from the RAAS to know if someone is sharing the ASP service.

The simplified UPnP communication paradigm between the smartphone and ASP is shown in Fig. 11. At the first step, the smartphone initiates a connection with the ASP by offering it the security profile. The security profile contains all parameters and credentials to establish a successful secure connection. Fig. 11 depicts the scenario where connection handshake is initiated by the smartphone. However, in our communication framework, either party can initiate the connection. After RATA successfully establish the connection, it notifies RADA. The RADA on both, the smartphone and the ASP issue request to get the remote CD and CP descriptions. At the next step, RADA announces the remote CD and CP to the local CP and CD, respectively. Thus, the smartphone CD and CP are updated about the ASP CP and CD, respectively and vice versa. Now at the final step, the smartphone CD and CP communicate with the ASP CP and CD, respectively in a normal way as if they are located in the same network.

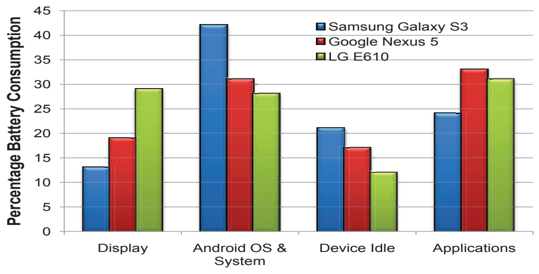


Figure 12. Applications power consumption on different devices.

VII. Expected Benefits

The expected benefits of the ASP service depend on the number of applications running on the smartphone and their dependence on the smartphone resources such as CPU, memory, WiFi, 3G/4G data etc. Further, the energy consumption also depends on the size of heartbeat messages and their periodic interval. Normally, multiple applications transmit heartbeat messages randomly, which forces the WiFi or 3G/4G data connection to stay always in the active state. Thus, the Internet interfaces normally constitute the major share of energy consumption. The energy consumption of same applications on different devices can be different based on the device built-in power management features provided by the manufacturer. These features may include reducing clock frequency and switching ON network interfaces periodically for short interval during idle periods (when backlight/LCD is OFF). For the sake of observations, we analyzed the battery consumption of three different daily used smartphones in Fig. 12 which are running multiple Internet-based applications (i.e., Google+, Facebook messenger, Whatsapp, Viber, Vonage, Voip, Skype, Dropbox etc). It can be observed in Fig. 12 that the energy consumption also depends on the device and its usage (e.g., display). However, the applications constitute a significant portion of overall battery consumption. Thus, reducing energy waste of background applications with the ASP service can significantly improve the battery life.

VIII. Conclusions & Future Work

Today, the smartphones have high memory and processing capabilities and are able to run concurrently multiple services and applications. However, most of the applications are Internet-based which periodically transmit/receive presence/heartbeat messages over WiFi or 3G/4G data connection. These applications significantly reduce the battery life of smartphone as they extensively utilize resources such as CPU, memory or keeping the Internet interfaces always in active state. In this paper, we proposed the ASP concept that suppresses/stops applications on the smartphone and maintains their presence on any other network device. The ASP returns the specific application control back to smartphone only when an event is received such as a new message. In this paper, we presented the requirements for the ASP framework and proposed different possible architectural designs. We have addressed different techniques that can be adopted to proxy the presence of applications on behalf of smartphones. Further, we have presented in details the design of our communication framework.

Indeed, the ASP concept can significantly improve the battery life of smartphones by reducing the energy waste due to background applications. Future work will focus on the implementation and evaluation of our proposed ASP framework.

References

[1] R. Palit, K. Naik, and A. Singh, "Impact of Packet Aggregation on Energy Consumption in Smartphones," in *7th International Wireless Communications and Mobile Computing Conference (IWCMC)*, 2011.

[2] T. Pering, Y. Agarwal, R. Gupta, and R. Want, "CoolSpots: Reducing the Power Consumption of Wireless Mobile Devices with Multiple Radio Interfaces," in *Proceedings of the 4th international conference on Mobile systems, applications and services*, 2006.

[3] G. Perrucci, F. Fitzek, G. Sasso, W. Kellerer, and J. Widmer, "On the Impact of 2G and 3G Network Usage for Mobile Phones' Battery Life," in *European Wireless Conference*, 2009.

[4] B. Aggarwal, P. Chitnis, A. Dey, K. Jain, V. Navda, V. Padmanabhan, R. Ramjee, A. Schulman, and N. Spring, "Stratus: Energy-Efficient Mobile Communication using Cloud Support," in *SIGCOMM '10*, 2010.

[5] R. Bolla, M. Giribaldi, R. Khan, and M. Repetto, "Design and Implementation of Cooperative Network Connectivity Proxy using Universal Plug and Play," in *10th FIA Book*, 2013.

[6] R. Bolla, M. Giribaldi, R. Khan, and M. Repetto, "Network Connectivity Proxy: An Optimal Strategy for Reducing Energy Waste in Network Edge Devices," in *The 24th Tyrrhenian International Workshop on Digital Communications*, 2013.

[7] G. Perrucci, F. Fitzek, and J. Widmer, "Survey on Energy Consumption Entities on the Smartphone Platform," in *IEEE 73rd Vehicular Technology Conference (VTC Spring)*, 2011.

[8] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha, "AppScope: Application Energy Metering Framework for Android Smartphone Using Kernel Activity Monitoring," in *Proceedings of USENIX*, 2012.

[9] K. Pentikousis, "In Search of Energy-Efficient Mobile Networking," in *IEEE Communications Magazine, Vol:48, Issue:1*, 2010.

[10] A. Rice and S. Hay, "Decomposing Power Measurements for Mobile Devices," in *IEEE PerCom*, 2010.

[11] M. Calder and M. Marina, "Batch Scheduling of Recurrent Applications for Energy Savings on Mobile Phones," in *7th Annual IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks (SECON)*, 2010.

[12] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications," in *Internet Measurement Conference (IMC)*, 2009.

[13] A. Sharma, V. Navda, R. Ramjee, V. N. Padmanabhan, and E. M. Belding, "Cool-Tether: Energy Efficient On-the-fly WiFi Hot-spots using Mobile Phones," in *ACM CoNext*, 2009.

[14] N. Ristanovic, J. Boudec, A. Chaintreau, and V. Erramilli, "Energy Efficient Offloading of 3G Networks," in *Proceedings of IEEE Eighth International Conference on Mobile Ad-Hoc and Sensor Systems, MASS '11*, 2011.

[15] K. Lee, I. Rhee, J. Lee, S. Chong, and Y. Yi, "Mobile Data Offloading: How Much Can WiFi Deliver?" in *IEEE/ACM Transactions on Networking, Vol:21, Issue:2*, 2013.

[16] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic Resource Allocation and Parallel Execution in the Cloud for Mobile Code Offloading," in *IEEE INFOCOM*, 2012.

[17] K. Kumar and Y. Lu, "Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?" in *IEEE Computer Vol:43, Issue:4*, 2010.

[18] Y. Agarwal, S. Hodges, R. Chandra, J. Scott, P. Bahl, and R. Gupta, "Somniloquy: Augmenting Network Interfaces to Reduce PC Energy Usage," in *6th ACM/USENIX Symp. On Networked Systems Design and Implementation (NSDI 09)*, Boston, MA, USA, April 2009.

[19] "UPnP forum, 2012. URL: http://www.upnp.org."

[20] "UPnP forum: Remote Access architecture 2, 2011. Available at URL: http://www.upnp.org/specs/ra/UPnP-ra-RAARchitecture-v2.pdf."