Analyzing Network and Energy Efficiency of Android Smartphone Applications

Rashi Garg
IIIT Delhi

Shaifali Gupta
IIIT Delhi

ABSTRACT
Mobile phones and tablets can be considered as the first in-carnation of the post-PC era. Their explosive adoption rates lead to the emergence of a variety of Applications. Most attractive feature of these applications is that they are available for a variety of OS. Many of these applications, for example VoIP clients, stay active in the background. In the background, they may not communicate large amounts of data. However, their regular bursts of activity can lead to large signaling overheads, wastage of radio resources, and draining of a phone's battery. Signaling overheads can lead to service outage by overloading the radio network controller of the 3G network. Thus in addition to aiding user, these convenience applications can pose several threats to the user handset and the Network. In this Work, We study the various Problems that persist affecting the user handset and Network and the solutions that exist for dealing with these problems. We attempt to classify the Problem and Solutions into various categories and list all the available solutions that exist for a given problem.

General Terms
Data and Information Systems, Mobile computing

Keywords
RRC states, Smartphone apps, resource utilization, Network efficiency, Energy Efficiency

1. INTRODUCTION
Universal Mobile Telecommunications System (UMTS) based 3G cellular networks have been widely deployed across the world. Riding on top of WCDMA technology, these networks promise to deliver a high quality service for data and voice communications. This, coupled with the advent of smart phones and mobile applications, has brought the Internet just a click away for modern users. On one hand, this easy access to the Internet through mobile phones has revolutionized the lives of end users, on the other hand, the continuous connectivity and resulting data traffic poses challenges to the performance of 3G networks. One major challenge is due to signaling overhead associated with the data-transfer. 3G wireless stacks use Radio Resource Control (RRC) protocol to handle control plane signaling between user

Equipment (UE) and UMTS Terrestrial Radio Access Network (UTRAN). According to RRC specifications, a radio channel is allocated to a UE only upon its request to establish connection for Data-transfer and is revoked after a certain period of inactivity. A large number of signaling messages are exchanged between the UE and access network during this process. A UE can either be in idle mode (IDLE) or in connected mode. In the connected mode the UE may be in the DCH or in the FACH state. The UE cannot send or receive data in IDLE. To do so, it needs to transition to one of the connected states. Transition to a connected state, see 1, is initiated when data buffers at the UE or for the UE in the network, exceed certain pre-configured thresholds. Amongst the connected states DCH consumes more energy and provides larger throughput to the UE. It is chosen over FACH for larger buffer occupancies. Transitions to lower energy states are initiated when buffer levels remain low (DCH->FACH) or if no data activity is detected (DCH/FACH->IDLE) for a certain timeout period. Even badly designed applications can cause damage unintentionally. Such applications not only increase the signaling load on the network but also cause battery drain of UE.

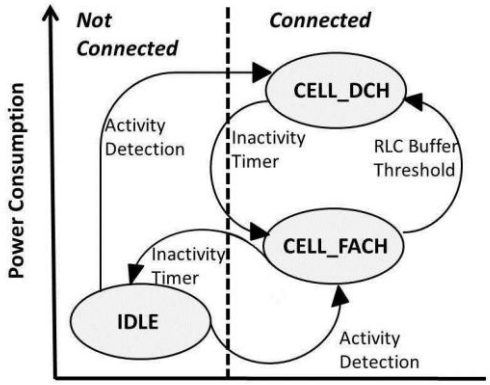

Figure 1: RRC State Machine (From: Android Phone Based Appraisal of App Behavior on Cell Networks [8],p.2)

Earlier the Battery Drain and Network Signaling problem was not given enough attention. But now scenario has changed. Due to emergence of variety of applications even the offline gaming applications are responsible for high resource usage. A recent study shows that 65-75% of the energy consumed in a gaming app (Angry Birds) on Android devices is spent by third party advertising modules [17]. There are various factors that are responsible for Battery Drain and Network Signaling and corresponding solutions are proposed to handle those factors effectively in order to minimize Battery Drain and Network Signaling as far as possible.

2. EFFECT ON USER HANDSET AND NETWORK
Apart from the battery drain and signaling there can be a range of problems that an application can create for the User Handset and the Network. Below is the list of issues that user handset and Network can face due to any application:

User Handset

1. Battery Life
2. Application Responsiveness
3. Background Data

Network

1. Signaling Volume
2. Data Volume transferred
3. Control Plan Usage
4. Radio resource Usage

3. PROBLEMS IN APPLICATION AFFECT-ING BATTERY AND NETWORK

There are a lot of factors responsible for affecting the Application performance and thus consequently affecting the User handset and the network. Authors in [17] observed that advertisements embedded in applications account for at-least 1% of Network Activity and in some cases this is the only network Activity. They found out that this is the problem in case of both Offline and Online Applications where the Offline Applications involve Gaming Applications also. A recent study showed that Majority of traffic in Angry Bird Gaming App is due to Advertisement that are embedded in the application and fetched when the user connects to the Internet. The authors in [13] listed out five different parameters that are responsible for Periodic Transfers thus preventing the device from going into idle mode and connecting to the network even for sending small amount of data.

Keep Alive Messages: These are used to prevent the TCP Connection from being closed by cellular NAT [13] [3]. When the Chatty Application like Facebook starts periodicity of these messages is 60 sec but in chatting mode it becomes even worse periodicity reduces to 20 sec. Thus causing signaling by sending small bytes of data and keeping the device always in active atate.

Measurements: These are used to collect data about the user experience and preferences and upload to data to a site with a periodicity of at least 1 minute. Researchers argue that periodicity should be increased to at least 2 or 3 minutes.

Polling: Polling is used in some applications for querying for new message with a lesser periodicity of 20 sec. This technique is even worse and causes high energy overhead than Push based Scheme that is used in certain apps like Facebook.

Advertisements: These are embedded in application and cause overhead while fetching ads and storing them in the device.

Periodic DNS Lookups: Although handsets have local DNS caches, some content providers may set DNS TTLs to be small for load balancing. Authors observed that IP addresses returned by periodic DNS lookups seldom change for the same host name, thus suggesting that DNS-based load balancing is not frequently performed although content providers have such capabilities.

A recent study showed that even the bugs in the application or OS can result in Battery Drain. Authors have listed various categories of bugs like App Bugs, System Bugs that were found to be responsible for battery drain and significantly affected Battery Life [12]. It was observed that in some cases after completion of transfer device still resides in the high power state DCH or FACH thus resulting in high energy overhead [6]. Authors in [2] [3] have recommended Fast Dormancy as a solution to reduce Battery Drain and increasing the Battery Life but it itself proves to be a problem since Fast Dormancy forces the device to enter into IDLE State immediately after completing the transfer. Thus for initiating a new transfer device will have to request for a connection thereby causing signaling overhead. To counter this, researchers have recommended few other solutions that are discussed in later sections. Authors in [18] have listed six different factors that can affect both the User Handset and Network-

Multiple Simultaneous TCP Connections: If an app opens multiple TCP connections, it reduces the bandwidth for all connections, reducing the throughput and decreasing performance.

Periodic Transfers: Server sends specific packet repeatedly where each transfer can incur a long tail. If the timing of these transfers is not regulated then it can result in high overhead in terms of Signaling and Battery Consumption.

Inefficient Technique for Opening Connection: Many applications do not handle connections efficiently. A common approach followed by the applications consists of initial bursts in the startup followed by a series of bursts.

Inefficient Technique for Closing Connection: Ideal case should be to close connection immediately after completing the transfer. But majority of Apps keep connections open even after they are no longer required.

Pinging Server on Screen Rotation: Many apps send a request to server on screen rotation even if there is no change in the data, thus wasting Network resources by causing unnecessary signaling and affecting the battery.

Duplicate content download:

A recent study showed that inefficient handling of Tail Time can pose serious issues. If it can be accurately predicted that there will be some transmission during tail time then connection can be maintained otherwise break the connection and save energy. If not accurately predicted and connection is maintained, this cause wastage of energy and radio resources. If there was a need of connection but prediction was different and connection was terminated then this will cause signaling overhead. Thus if the Tail Time is handled carefully it can be used to avoid unnecessary overhead in terms of Energy Consumption and Network Signaling [10] [14]. Authors in [4] observed that Multiple Applications requesting for connection can increase signaling load considerably. Applications on smart phones periodically connect and disconnect to/from the network for updates. Each connection/disconnection attempt requires several message exchanges between the smart phone and the network. All these message exchanges generate signaling load on the network. If many such applications request for connection at the same time then it can lead to a large amount of signaling. In [5], authors discussed that increased usage of signaling resources have a significant impact on battery life and reduces the perceived quality of devices on the other hand it can result in congestion in the network thus leading to service accessibility problems.

Recent work by Kononen et. al. [11] showed that timer requirement are not strict so if the timer is handled carefully it can be used to decrease the consumption of network re-sources. Gadekar et al. [7] discovered that gaming application utilizes maximum processor power capabilities and maximum utilization of heap. Authors have considered 2 problems in the gaming application scenario that can affect the Battery life and the Network. Stack Overflow: In Smartphone environment where several apps are running it is quit essential to handle Stack efficiently. If not handled carefully it can result in app crashing and Battery issues and the other one is Advertisements Overhead [7] [17] [13]. Siekkinen et al. [15] have found that streaming apps are more power hungry due to radio communication, computation and display and multimedia presentation. Thus efficient handling of streaming traffic is necessary.

4. SOLUTIONS THAT CAN INCREASE THE BATTERY LIFE AND IMPROVE NETWORK SIGNALING

Caching Advertisements:

Basic design of Ad-Caches is based on Batching and Caching. Ad-Cache monitors the network conditions of the mobile interface (i.e., signal strength and type of network). This is done in order to temporarily defer the update if the network conditions are not ideal. The mobile agent is a continuously running background service that prefetches ads into a data structure and serves them locally on requests by apps at the minimum power cost. The data structure is persistent (even when phone reboots), and is filled completely when the agent is initially started with a set of ads specified by the ad network and is then updated periodically. Ads are prefetched based on their Time to Live (TTL) value, defined by the ad network to remove outdated or invalid ads. The mobile agent acts as the coordination point for delivering ads to apps installed on the device, making sure that no single app requests ads excessively and that radio access is given consciously.

This design has following affects:

1. There are no cache misses
2. Ads will not be pre-fetched if there are enough valid ads or there is no app demand hence reducing traffic volume
3. Allows AdCache to efficiently use network and energy-intense resources by exploiting batching techniques

Dealing with Periodic Transfers

Authors in [13] have proposed various techniques for dealing with periodic transfers.

1. Piggybacking: Target transfers (Periodic Transfers) can be shifted earlier, or be postponed till later, so that they can potentially be overlapped with non-target transfers, thus reducing the tail time, where no-target Transfers denote the transfers that cannot be controlled i.e. User initiated Transfer.
2. Batching: To reduce the tail time, several periodic transfer can be grouped into a single bursts. But this technique has a disadvantage that it can result in increase in periodicity.
3. Fast Dormancy: In this method Handset requests for an immediate transition to the IDLE state by sending RRC Control Message instead of waiting for the Inactivity Timer to expire. This method reduces the tail time But it has an inherent limitation it results in increase in signaling load.
4. TailEnder: This tool schedules the transfer by delaying them and sending them together. This results in reduction in energy consumption. It is similar to Piggy-backing with a difference that TailEnder can only delay transmissions; it cannot send them ahead of schedule.

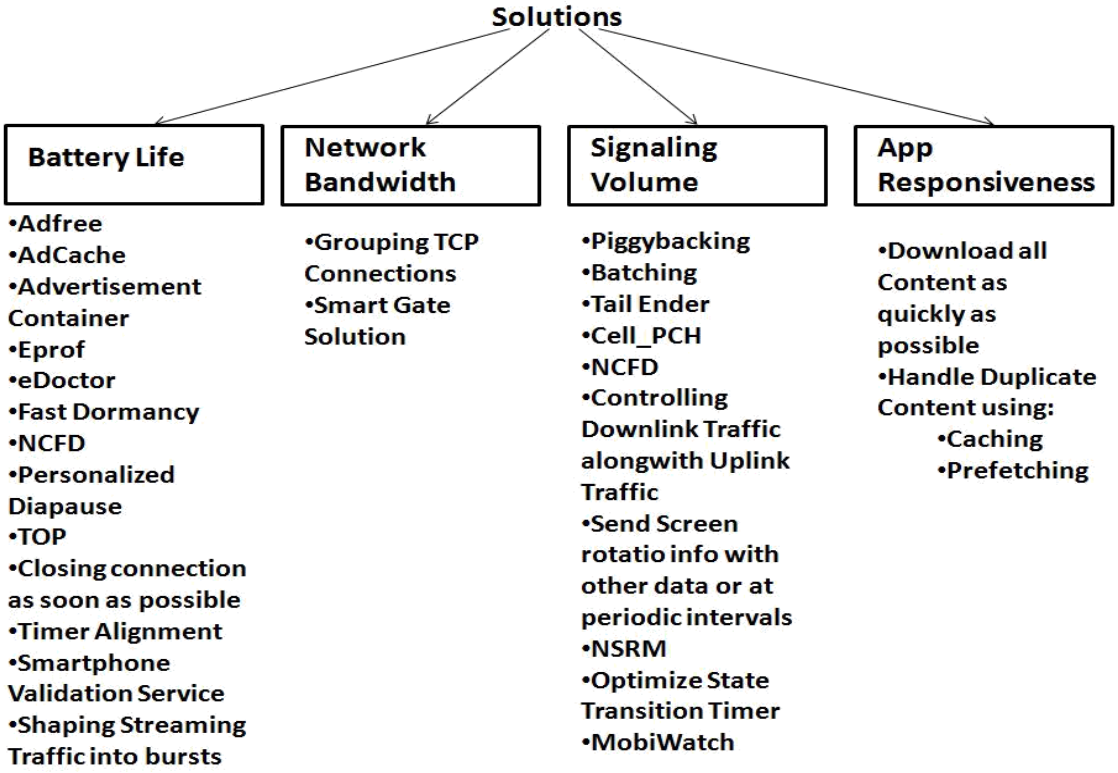

Figure 2: Classification of Solutions

Dealing with Abnormal Battery drain

Xiao et al. [12] designed a tool eDoctor to deal with Abnormal Battery Drain Issue. Authors have mentioned Energy profilers like Eprof can also be used, but they have major limitation that they can not differentiate between normal and abnormal battery drain.

eDoctor:
eDoctor records resource usage and relevant events, and then uses this information to diagnose issues and suggest resolutions. eDoctor uses phases to capture apps' time varying behaviors. It then identifies suspicious apps that have significant phase behavior changes. eDoctor also records events such as app installation and upgrades, configuration changes, etc. It uses this information in combination with anomaly detection to pinpoint the culprit app and the causing event, as well as to suggest the best repair solution.

In [6], Authors have proposed a solution for minimizing the tail energy. Authors have designed TailEnder, a protocol that reduces energy consumption of common mobile applications. Applications that can tolerate small delay TailEnder schedules the transfer in such a way that cumulative energy is minimized at the same time meeting the user specified deadlines. TailEnder is based on two techniques and uses the one that is most appropriate for an application.

1. Batching: This technique is use for applications like Email, News feed i.e. the category of application that can tolerate some delay.

2. Prefetching: For applications like web search engines that can benefit from prefetching, TailEnder determines what data to prefetch so as to minimize the overall energy consumed. Prefetching useful data reduces the number of transfers and their associated cumulative tail energy, while prefetching useless data incurs additional transmission energy. TailEnder uses a probabilistic strategy to balance these concerns

Dealing with Network Signaling

Authors in [2] [1] have mentioned three solutions for dealing with Battery issue and Network Signaling.

1. Fast Dormancy: Using this solution handset request for immediate transition to IDLE state by sending RRC Control message. This method is successful in reducing the energy consumption but at the same time it results in increase in Network Signaling, as the device will have to send the entire series of RRC Control message again to initiate the connection.

2. PCH: In this handset enters into PCH state (the intermediate state between IDLE and FACH) after completing transfer. Main advantage of handset entering into PCH is that it requires only twelve signaling message for the round-trip i.e. from PCH to Active State and back to PCH in contrast to IDLE that requires 30 signaling message for the round-trip. Thus PCH reduces the Network Signaling and at the same time also contributes in reducing the energy consumption.

3. Network Controlled Fast Dormancy: It is also known as Release 8 Fast Dormancy. With Network Controlled Fast Dormancy, handsets can act dynamically exploiting PCH on the part of Network supporting PCH, and using Fast Dormancy on the part of network enabled with only Fast Dormancy.

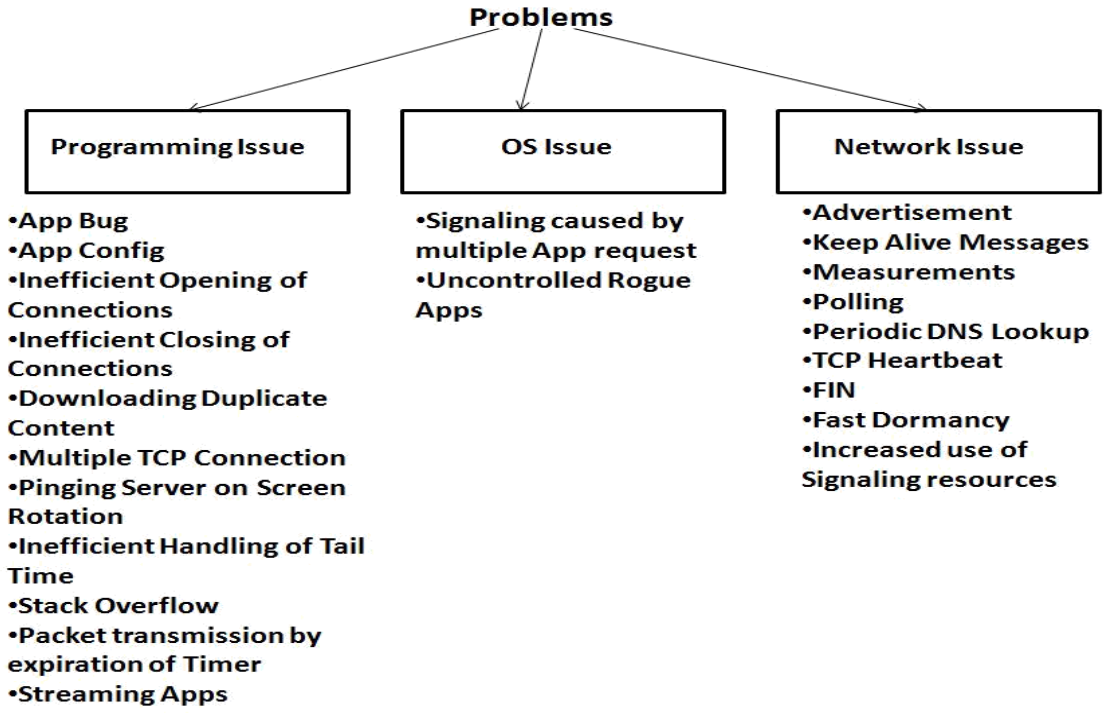

Figure 3: Classification of Problems

Dealing with Bugs in App design or OS

To counter TCP heartbeats and FIN, downlink traffic should be controlled along with uplink traffic. In [18], Authors have listed out solutions that can be possible for the problems that are listed in Section-3.

1. Multiple TCP Connections can be grouped together from the same server, thus saving energy and reducing the response time by effectively managing multiple persistent connections.

2. One solution for Periodic Transfer is to increase the periodicity between the transfers and other can be piggybacking the transfers.

3. In case of inefficient opening of connections, one approach is to download all the content needed by the app as quickly as possible. Grouping of data allows the faster download thus reducing the energy consumption.

4. Inefficient closing of connections can be handled by closing all the connections after transmitting all the data.

5. Authors suggested that information regarding change in screen orientation should be sent along with other data at scheduled intervals rather than opening a separate connection every time when there is a change in screen orientation.

6. Duplicate Content problem can be solved either using caching or prefetching.

Dealing with Tail Time

Yeseong et al. [10] has designed a tool Personalized Di-apause for efficient handling of Tail Time. It is based on novel automatic extraction of meaningful network activities into network context blocks. Personalized Diapause takes advantage of personalized network context usages in deciding when to release a radio connection within the tail time. It then predicts user level network activities from running app and does not depend upon app developers for the predictions.

Authors in [14] have designed a tool TOP (Tail Optimization Protocol). It requires no change in cellular infrastructure and only minimal changes in smartphone applications. It leverages the knowledge of applications that predict the idle period after each data transfer. It was observed that TOP saved the overall radio energy (up to 17%) and radio resources (up to 14%) by reducing tail times by up to 60%. Both of these employ Fast Dormancy to notify network for immediate release of radio resources.

Authors in [4] proposed a new technique NSRM (Network Socket Request Manager) to manage the network traffic. NSRM gates application requests on smart phones when they are in background mode. By gating only background mode requests, NSRM does not affect user experience. It does not gate requests when handset is in active mode. Gates can be either opened immediately when the handset becomes active or at periodic interval. When the gate is opened, all the pending requests are bundled into a batch and sent over the network which also reduces the number of connection attempts, thus reducing the network signaling. NSRM can be used effectively for applications like news, social networking, weather etc. that generate requests even when the device is in background.

Dealing with Fast Dormancy and Tail time

To deal with Fast Dormancy and keep Alive messages one of the ways can be to use Release 8 Fast Dormancy [3]. State Transition Timer can be optimized in order to achieve a balance between the Battery life and Network Signaling [3].

Other interesting solutions

In order to deal with increased signaling resources and uncontrolled rogue apps, authors in [5] discussed two solutions. One of the solutions is the Smartphone Validation Service in which the operator benchmark the new devices, OS versions and apps. Data collected from network and device logs can be used to generate key performance indicators and reports to high-light any changes in behavior and their potential impacts.
Second Solution is Smart Gate Solution in which operators manage how and when device connects and transfer data on the network, thus providing operator with the full control of the smartphone behavior.
Ville et al. in [11] provided a solution for the Timer Alignment. Their approach involves the use of both minimum and maximum threshold of timer for the timer alignment where as in second approach only maximum threshold of timer is available.
Amar et al. in [7] developed a Rubik's cube game and analyzed the issues discussed in Section 3. They have developed a advertisement container to counter all the three issues. advertisement container is used to download and update the stack of ads. These ads get compressed and displayed in low resolution so they do not affect the Battery adversely and they will access minimum area of heap without affecting the performance of the application.
One of the solutions for the Streaming application like Pandora has been suggested by Authors in [15]. Authors suggested that shaping the streaming traffic into burst will give device time to switch from high power active state to low power state.

5. SOLUTION CLASSIFICATION

Solutions for the various problems can be classified into four categories on the basis of problem they cater to. Figure 2 shows detailed the classification.

Battery Life: Solutions falling into this category cater Battery Life Problem encountered in user handset. Solutions like Advertisement Container, Fast Dormancy target this problem.

Network Bandwidth: Network Bandwidth can be managed using Smart Gate Solution or Grouping Multiple TCP Connections into one.

Signaling Volume: Solutions like Piggybacking, TailEnder, and Batching are more effective in solving this problem.

App Responsiveness: This category includes solutions like Handling Duplicate Content using Caching, Prefetching.

6. PROBLEM CLASSIFICATION

Problems can be classified into three categories:

Programming Issue: Problems in this category occur mainly due to error in Programming from the developer side. Programming mistake is responsible for various issues like Battery Drain, increased Networks Signaling. Problems like App Bug, Inefficient opening and Closing of connections can

be categorized into this category.

OS Issue: Problems like Uncontrolled Rogue Apps, Multiple App Request can be grouped under this category where Problem persists at the OS Level.

Network Issue: Problem exists at the Network level. Advertisement, Keep-Alive Messages, FIN, Fast Dormancy are the examples of the problems that can be categorized into this category.

Figure 3 shows the classification of Problems.

7. CONCLUSION

The tools proposed by different researchers can be used by Application Developers to design better applications that are more aware of the energy consumption on the handset and network usage. Mobile Network Operators can use these tools to identify the most resource intensive applications and estimate the technique using which the network behavior of application can be improved. Thus we can conclude that list of these problems and solutions can help Application developer to get an insight into the details of the issue and to see whether the Problem is at Network level, OS level or Application level, thus helping in selection of better solution.

Table 1: Table showing list of Problems and available solution

Problem	**Solution**
Advertisements	Adfree Blocker App AdCache Tool Advertisment Container
Periodic Transfers	Piggybacking Batching Fast Dormancy TailEnder
App Bug App Config	Energy Profilers – Eprof eDoctor
High Tail Energy	TailEnder
High Energy Consumption and Signaling	PCH Fast Dormancy Network Controlled Fast Dormancy
TCP Heartbeats FIN	Control Downlink Traffic
Multiple TCP Connections Periodic Transfers Inefficient technique for opening connection Inefficient technique for closing connection Pinging server on screen rotation Duplicate content download	Grouping Connections from same server Piggybacking or increasing time between transfers Download content needed by app as fast as possible Closing all connections immediately after data transfer Sending Device Orientation data along with other data at scheduled intervals Caching Prefetching
Tail Time	Personalized Diapause TOP
Multiple applications requesting for connection	Network Socket Request Manager
Fast Dormancy Keep Alive Messages	Rel 8 Fast Dormancy Optimization of State Transition Timer
Increased use of signaling resources Uncontrolled rogue apps	Smartphone Validation Service Smart Gate Solution
Timer	Timer Alignment
Advertisement overhead Battery Drain Stack Overflow	Advertisement Container
Streaming Traffic	Shaping Streaming Traffic into Bursts

8. REFERENCES

[1] Behavior analysis of smartphone. Huawei Technologies Co. Ltd., 2011.

[2] Understanding smartphone behavior in the network. Nokia Siemens Networks Smart Labs, 2011.

[3] Behavior analysis of smartphones in 3g networks. Aexio, 2012.

[4] Managing background data traffic in mobile devices. Qualcomm Incorporated, 2012.

[5] Understanding and managing smartphones in the network. Aspire Technology Ltd., 2013.

[6] A.N. Balasubramanian, A. Balasubramanian, and Venkataramani. Energy consumption in mobile phones: A measurement study and implications for network applications. ACM IMC, November 2009.

[7] A. C. Gadekar, A. U. Jagtap, A. J. Kunchalas, and A. Jadhav. Rubik's cube application for android mobile phones with addressing android operating system problems. Global Journal of Computer Science and Technology, 12(0975-4350), April 2012.

[8] S. Gupta, R. Garg, N. Jain, V. Naik, and S. Kaul. Android phone based appraisal of app behavior on cell networks. Technical Report IIITD-TR-2013-003, IIIT-Delhi, October 2013.

[9] H. Holma and A.Toskala. WCDMA for UMTS: Radio Access for Third Generation Mobile Communications. John Wiley and Sons Inc., New York,USA, 2004.

[10] Y. Kim and J. Kim. Personalized diapause: Reducing radio energy consumption of smartphones by network-context aware dormancy predictions. ACM IMC, October 2012.

[11] V. Konnen and P. Paakkonen. Optimizing power consumption of always-on applications based on timer alignment. IEEE, January 2011.

[12] X. Ma, P. Huang, X. Jin, P. Wang, S. Park, D. Shen, Y. Zhou, L. K. Saul, and G. M. Voelker. edoctor: Automatically diagnosing abnormal battery drain issues on smartphones. USENIX Association Berkeley, CA, USA c 2013, April 2004.

[13] F. Qian, Z. Wang, Y. Gao, J. Huang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Periodic transfers in mobile applications: Network-wide origin, impact, and optimization. ACM WWW, April 2012.

[14] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Top: Tail optimization protocol for cellular radio resource allocation. IEEE INCP International Conference, October 2010.

[15] M. Siekkinen, M. Ashraful, and M. Aalto. Streaming over 3g and lte: How to save smartphone energy in radio access network-friendly way. ACM Movid, february 2013.

[16] M. Paolini and S. Fili. The taming of the app. Sponsored By:Seven Networks(http://www.seven.com/), 2013

[17] N. Vallina-Rodriguez, J. Shah, A. Finamore, Y. Grunenberger, H. Haddadi, K. Papagiannaki, and J. Crowcroft. Breaking for commercials: Characterizing mobile advertising. ACM IMC, November 2012.

[18] B. Weir and D. Sillars. Top radio resource issues in mobile application development. AT&T Developer Support Team, 2011.