2018-03-05

Reducing Energy Consumption and Latency of Applications on Wireless Devices

Sehati, Ali

UNIVERSITY OF CALGARY

Reducing Energy Consumption and Latency of Applications on Wireless Devices

by

Ali Sehati

A THESIS

SUBMITTED TO THE FACULTY OF GRADUATE STUDIES

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF DOCTOR OF PHILOSOPHY

GRADUATE PROGRAM IN COMPUTER SCIENCE

CALGARY, ALBERTA

March, 2018

Abstract

Energy consumption and delay are the key factors influencing users' quality of experience for applications running on wireless devices. In cellular networks, providing a satisfactory user experience faces several challenges caused by the poor interactions between multiple factors. First, cellular networks suffer from long round trip times and also employ a radio resource control protocol, which keeps the radio of the device in high power state even after completion of a data transfer. Second, in most applications, completing a user action involves many round trips over the high-latency cellular link, which can lead to poor application performance. Moreover, periodic and intermittent traffic pattern exhibited by the majority of applications can result in serious energy inefficiencies. In this thesis, we address these challenges from the network and end device perspectives. First, we design a network-centric solution, called WebPro, that adopts speculative loading and bundling techniques to reduce latency and energy consumption of mobile web browsing. Performance evaluation results obtained through live experiments indicate that WebPro outperforms state-of-the-art, though the degree of improvement varies for different webpages.

Then, we focus on energy-delay tradeoff on end devices and design algorithms to balance the energy-delay tradeoff inherent in bundling. Specifically, we formulate a generalized notion of bundling as an online optimization problem. The objective of this problem is to minimize the bundling cost defined as a weighted summation of energy and delay costs. Based on two different energy cost models associated with smartphones and internet of things (IoT) devices, we develop online algorithms to solve the optimization problem. A distinctive feature of our online algorithms is that they do not rely on any assumption about the traffic pattern or nature of applications. We provide theoretical performance bounds for our proposed algorithms by comparing them to an optimal offline algorithm. We evaluate the performance of our algorithms in a range of realistic scenarios using both model-driven simulations and

real experiments on a smartphone. Our results show that depending on the delay tolerance level of a user, energy savings ranging from zero to about 100% can be achieved using our algorithms.

Acknowledgements

I would like to thank my supervisor Dr. Majid Ghaderi. I appreciate the time he spent discussing research ideas with me during our weekly meetings. His comments on the drafts of my work helped me improve the quality of my writing and research. Without his guidance and support, this thesis would hardly have been completed.

I express my warmest gratitude to members of my examination committee, Dr. Mea Wang, Dr. Diwakar Krishanmurthy, Dr. Geoffrey Messier, and Dr. Majid Khabbazian for their time to read this thesis and their thoughtful feedback. Their comments and suggestions enhanced the quality of this thesis. I am also grateful to Martin Arlitt for his help on various occasions during my PhD.

I should mention the staff of the main office of the Department of Computer Science. Especially, I would like to thank Britta Hicks and Jackie Hunt who helped me with my scholarship applications and defense process.

I appreciate the financial support that I received from the University of Calgary and the Government of Alberta in the form of scholarships, teaching and research assistantships.

I would like to thank many friends and fellow graduate students at the University of Calgary who made my life happier during my stay in Calgary. I am deeply indebted to my parents for their perpetual support and unconditional love. They have been the main source of motivation and encouragement during my studies.

Table of Contents

List of Tables

List of Figures and Illustrations

List of Symbols, Abbreviations and Nomenclature

Symbol	Definition
AJAX	Asynchronous JavaScript and XML
BE	Break-Even Algorithm
BLE	Bluetooth Low Energy
CDF	Cumulative Distribution Function
CDN	Content Distribution Network
CR	Competitive Ratio
CV	Coefficient of Variation
DOM	Document Object Model
DRX	Discontinuous Reception
DynAck	Dynamic TCP Acknowledgement Problem
eNB	Evolved NodeB
EnerB	Energy-Aware Bundling Problem
HOL	Head of Line
HTTP	HyperText Transfer Protocol
IoT	Internet of Things
MPTCP	Multi-Path TCP
OPT	Optimal Offline Algorithm
PBB	Proxy Based Browsing
PDCCH	Physical Downlink Control Channel
PLT	Page Load Time
RRC	Radio Resource Control
RTT	Round Trip Time
TTR	Tail Time Ratio

UE	User Equipment
URL	Unified Resource Locator
VPN	Virtual Private Network

Chapter 1

Introduction

1.1 Motivation

Recent years have witnessed an unprecedented increase in global mobile data traffic. According to Cisco VNI report, it is estimated that the overall mobile data traffic will grow to 49 exabytes per month by 2021, which is a sevenfold increase compared to 2016 [2]. The ever increasing number of wireless devices with different capabilities is one of the main reasons for such a growth in data traffic. It is expected that by 2021, there will be about 11.6 billion wireless devices worldwide [2]. This is, in part, due to the rise of smartphones and tablets, which have become an integral part of our daily lives. The emergence of new technologies such as wearable and Internet of things (IoT) devices also contributes to this growing trend.

The surging popularity of wireless devices is due to the development of an increasing number of diverse applications that impact people's everyday lives. These applications provide a wide range of services such as social networking (e.g., Facebook), entertainment, location-based services (e.g., Google Map), and environmental monitoring in the context of IoT. The growing interest in such applications is evidenced by the fact that there are over 1 million apps hosted on Android's Google play store [3]. It is important to provide the users of these applications with a satisfactory experience. *Delay* and *energy consumption* of applications are two essential factors that affect user satisfaction:

- Users of most applications are sensitive to delay. For example, in the context of mobile web browsing, recent studies show that users' expectations of the page load time are increasing and it is likely that people will demand sub one-second load times in the near future [4]. Also there have been several studies that reveal the ill-effects of high application latency on business profits. As an example, it is reported that 57% of online shoppers will

wait at most three seconds before switching to a competitor site, which could result in lost revenues [5]. Application latency will be even more important in the near future, considering the growing interest of users in applications such as video streaming and cloud-based gaming where there exist stringent requirements on latency [4].

- On the other hand, wireless devices derive the required energy for their operation from batteries with limited capacities. Despite dramatic improvements in other hardware components of wireless devices, battery technology evolves at a slower rate which makes battery life a major bottleneck for satisfactory user experience. For instance, in the time period between 2012 and 2015, the average capacity of smartphone batteries has only increased by 500 mAh [6]. Battery life is even more important in the context of IoT, considering the prevalence of scenarios targeting multi-year lifetimes and also the fact that the cost of replacing batteries is often a major operational expenditure [7].

Any solution that aims to improve the quality of experience for application users (in terms of energy and delay) should consider the connectivity choice of wireless devices. The reason is that the operation of a majority of applications depends on a reliable Internet connection so that they can communicate with remote servers (possibly hosted on cloud platforms). The wide deployment of cellular networks, enables near-constant Internet connection for wireless devices. As a result, cellular networks have become the main connectivity choice for wireless devices operating in wide area networks. However, in cellular networks, providing a satisfactory experience for application users faces new challenges that did not exist for traditional desktop computers and their software. *These challenges stem from the co-existence of two main factors and the interplay between them. The first factor is the unique characteristics of cellular networks. The second factor is the nature of the interactions between applications and their corresponding remote servers.* We will discuss these factors in the following sections.

1.2 Characteristics of cellular networks

Long round trip times:

It has been reported that compared to wired and WiFi networks, cellular networks have longer transmission and access delays [8]. In particular, the wireless hop in a cellular network has longer round-trip times (RTT) (of the order of 70-86 msec [9]), which leads to higher values for end-to-end RTT. Such long RTTs are due to dynamic conditions in wireless channels (caused by fading, shadowing, interference and mobility of wireless devices) as well as the radio access mechanisms in cellular networks that involve several rounds of signalling exchange between the device and the network. Because of long RTTs, applications experience higher delays over cellular networks which can be frustrating [8]. For example, a recent study on the load times of the top 200 websites indicates that the 80^{th} percentile load time on a smartphone using a 4G connection is more than 15 seconds [10]. While long RTTs of cellular networks are the primary culprits for high load times, complex interactions between computation activities (parsing, script evaluation, rendering) and network transfers during the page load process also contribute to such high latencies. Thus, there is a need for new solutions to reduce the latency experienced by applications over cellular networks.

Radio resource control protocol:

Wireless devices use cellular radio interfaces to communicate over cellular networks. One of the characteristics of the radio interface is that it consumes a lot of power and is responsible for a significant portion of the total device power usage [11]. Recent studies show that a significant portion of the radio energy is spent when there is no active data transfer [12, 13]. This stems from a resource management policy known as the radio resource control (RRC) protocol, which is employed by cellular networks to efficiently utilize their limited radio resources.

Specifically, under RRC protocol, a cellular network associates a state machine with each wireless device to control the radio interface of the device. The state machine prevents the

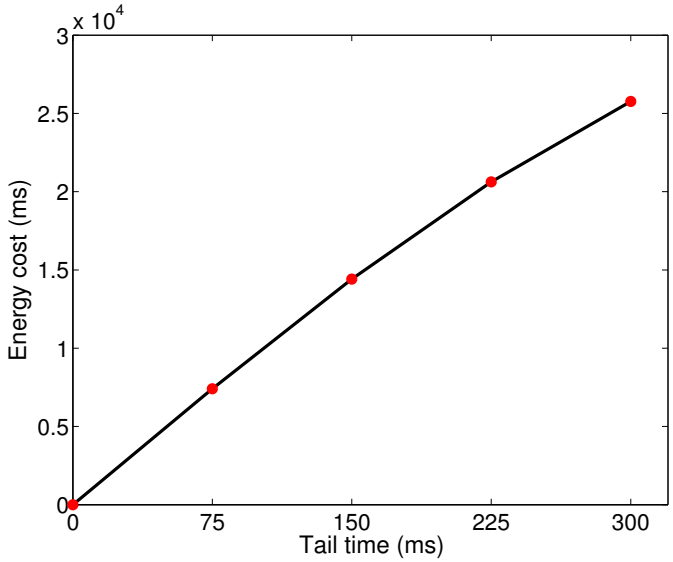


Figure 1.1: Effect of tail time on energy consumption

radio of the device from immediate demotion to the low power state once a data transfer is completed. Instead, the radio of the device stays in high power state until the expiration of a timer called *tail time*. When there is no network activity for the duration of the tail time, the radio switches to the low power state. The main reason for this behavior is to avoid switching radio states in case any data transfer request arrives during the tail time, which in turn reduces network signalling overhead. Tail time also helps to alleviate other inefficiencies resulting from frequent state switches. In particular, other than causing signalling overhead, switching radio states also takes time and consumes energy [12]. In 3G/LTE networks, typical values for the tail time are around several seconds depending on the carrier configuration [9, 14].

We performed a simulation experiment in order to gain a better insight on the effect of the tail time on radio energy consumption. We generated 100 random data transfer requests where inter-arrival times are sampled from a normal distribution with mean 300 ms and standard deviation 150 ms. Consider a setting where requests are granted as soon as they arrive. Fig. 1.1 depicts the amount of time the radio spends in the high power state (energy cost) under different values of the tail time. It can be seen that when the tail time is zero,

energy consumption due to tail effect is zero. However, by increasing the tail time, the amount of energy spent during tail periods increases.

It has been shown that poor interactions between the traffic patterns of applications and the RRC state machine, can negatively affect the energy efficiency of the applications and quickly drain the battery of the device [15]. As a result, reducing energy consumption of the applications requires optimization strategies that are aware of the cellular RRC state machine. A well-known technique to alleviate the effect of tail energy is *request bundling* [12, 13, 16–20]. With bundling, rather than granting individual data transfer requests as they arrive, multiple requests are bundled together and granted at once. Thus, the tail energy is amortized over multiple transfers in a bundle, which results in reduced radio energy consumption.

1.3 Interactions between applications and servers

As mentioned before, the interactions between applications and servers and specifically, the resulting network access pattern (referred to as traffic pattern) can create challenges for ensuring a satisfactory user experience in cellular networks. Also notice that there exists a diverse set of applications and different applications follow different patterns in accessing the network. As a result, we focus on some particular classes of well-known network traffic patterns:

Traffic from browsing applications:

The first class of traffic patterns that we consider is the one associated with the browsing applications. Web browsing is one of the core applications on wireless devices (e.g., smartphones and tablets) and is responsible for more cellular traffic than any other application, except for multimedia streaming [21]. There are also many applications that are simply customized programmable browsers (such as news applications) and hence follow the same behavior of regular web browsers [22]. Web browsers are used to load web pages, which

typically comprise of hundreds of web objects hosted on multiple server domains.

A main characteristic of the page loading process is that local computations such as HTML parsing and script evaluations are interleaved with network transfers [23]. Such a process creates a traffic pattern that is characterized by a large number of object downloads with idle gaps between them (created because of local computations). The reason for this behavior is that when a user requests a page, the web browser has no knowledge about the required resources of that page. So, the browser first downloads the base HTML file of the page and then parses it to identify the list of the objects referenced in the page. After parsing the base HTML file, a limited number of parallel TCP connections are initiated to fetch the identified objects. It is also possible that some of the embedded objects (script files such as JavaScript and CSS) themselves, will refer to other objects. Thus, script files should be evaluated and the result of their evaluation could reveal references to other embedded objects.

Put another way, because of the inter-dependencies between web objects, downloading all the required objects of a page may take multiple round trip times. This can lead to unsatisfactorily high page load times on wireless devices that operate in cellular networks with high RTT values.

Periodic and intermittent transfers:

Another well-known traffic pattern in wireless devices is the periodic and intermittent transfers where applications periodically and/or intermittently send or receive some amount of data to/from a remote server [15]. Such traffic pattern is indeed a prevalent pattern among wireless devices. This pattern is created because of activities such as data syncs and status updates [24], polling, keep-alive messages for push-based services [25], user-behavior measurements, computation offloading to the cloud [16], and advertisement transfers [15]. As described earlier, web browsing also involves downloading a large number of objects where there are idle gaps between consecutive web object downloads [26,27]. As a result, the traffic

generated by web browsers can also be considered a specific case of this pattern. Finally, periodic transmission of messages is a characteristic feature of the IoT applications as well [28]. Such a behavior happens when an IoT device regularly transmits status updates to a central server. For example, this is a typical behavior in smart meter reading (e.g., gas, electricity, and water) scenarios. As another example, IoT devices may send location updates to a central server, on a regular basis [29].

It is well-known that periodic and intermittent data transfers are highly energy inefficient. This is explained by the characteristics of the RRC state machine described above. Specifically, in every instance of a periodic transmission, an additional tail energy is consumed by the cellular radio interface. Therefore, periodic transfers can keep the radio in high power state for a long period of time, leading to rapid depletion of the device battery.

1.4 Thesis objectives

In this thesis we aim to reduce the latency and energy consumption of applications running on wireless devices. We consider wireless devices that are connected to the Internet through a cellular network. Based on this, three problems are studied in this thesis, namely:

- **Reducing latency and energy of mobile web browsing:** Due to the popularity of mobile web browsing, we investigate reducing latency and energy consumption of web browsing on mobile devices (e.g., smartphones and tablets). As mentioned before, the first step in loading a web page is fetching the base HTML file in order to discover the list of the referenced objects in a page. *This process takes roughly one round trip time and constitutes a significant portion of the web browsing delay on wireless devices as cellular networks suffer from long transmission and access delays.* In an effort to reduce the latency of mobile web browsing, we aim to eliminate this initial delay.

  On the other hand, as mentioned before, the page load process involves a large number of object downloads that happen intermittently with idle gaps between them. Such a network

access pattern can keep the radio of the device in high power state for longer durations, which can quickly drain the battery of the device. Thus, we study solutions that in addition to reducing the page load time, are able to reduce mobile energy consumption as well. We use *speculative loading* technique to reduce page load time in our proposed solution. Moreover, *bundling* technique is employed to reduce energy consumption during the page load process. Also in contrast to the prior attempts, we target a solution that is transparent to the end systems (client and server side), does not require modifying HTTP protocol and is well suited for web browsing on mobile devices.

- **Balancing energy-delay tradeoff for request bundling on smartphones:** For the first problem, we propose a solution that packs all the required objects in one bundle and achieves a fixed point on the energy-delay tradeoff associated with bundling (characterized by maximum energy saving). In order to be able to cover the entire spectrum of the energy-delay tradeoff, we consider a generalized notion of bundling (called request bundling) and formalize the energy-delay tradeoff that results from it. Notice that request bundling on smartphones can reduce radio energy consumption not only in mobile web browsing, but in a wide range of scenarios characterized by periodic and intermittent data transfer requests. For instance, it has been shown that request bundling can reduce radio energy consumption in computation offloading [16], and also delay tolerant applications [13].

  As mentioned before, bundling reduces radio energy consumption as it consolidates multiple tail energies into one tail energy. A critical question when implementing request bundling is *how long to wait to create a bundle*. By waiting too long, potentially more requests can be bundled together leading to more energy savings. This comes, however, at the expense of increased delay, which may negatively affect the performance of applications, and consequently the user experience.

  Clearly, with bundling, there is a tradeoff between energy saving and increased delay. In general, the energy-delay tradeoff depends on various contextual and technological

factors [30]. For example, a user possessing a smartphone with full battery may prefer earlier task completion over energy saving. As another example, Android smartphones allow users to specify their preference for energy saving over latency reduction by enabling the *battery saver mode*, which reduces device's performance by limiting location services and preventing applications from fetching new data in the background. However, given that users value both energy efficiency and low latency, it is desirable to devise solutions that have the flexibility in achieving different energy-delay tradeoffs. To this end, we study general and systematic solutions to balance the energy-delay tradeoff.

Specifically, we study a bundling algorithm that can minimize the bundling cost defined as a weighted summation of energy and delay costs. Energy cost is modeled using an On/Off model, which measures the time the radio spends in the on state. The exact definition of the delay cost will be clarified later in Chapter 4. Preference over delay versus energy is controlled by including a weight factor in the cost function. The difficulty in designing such a bundling algorithm is that the bundling decisions have to be made online *without knowing the timing of future data transfer requests*.

- **Online energy management in IoT applications:** As mentioned before, periodic transmission of small messages is a characteristic feature of IoT applications as well [28]. On the other hand, wide deployment of 4G cellular networks based on LTE has made LTE a natural candidate for IoT connectivity [31]. Consequently, the high energy inefficiency resulting from periodic transmission of small messages over LTE could be detrimental to the limited battery life of IoT devices. This energy inefficiency is due to the fact that every time a small message is transmitted, an additional tail energy is consumed by the LTE radio. In contrast to large messages on smartphones, when transmitting small messages on IoT devices, the tail energy is significant compared to the energy consumed for transmitting the message itself.

  To ensure ultra-long battery life for IoT devices, recently, 3GPP has proposed a number

of LTE enhancements for low-power wide area communications including Machine Type Communication in LTE-M specifications [32, 33]. To reduce power consumption, LTE-M includes several power saving mechanisms such as the extended Discontinuous Reception (DRX), which enables an IoT device to manage its LTE radio more efficiently by minimizing the time the radio spends in high power active mode [34]. Specifically, an intermediate power state, called DRX state, is introduced in the RRC state machine, where the wireless device monitors physical control channel less frequently.

Although the proposed power saving mechanisms are effective in reducing LTE radio energy consumption, they are oblivious to the specific requirements of different IoT applications in terms of delay and energy. In particular, while different IoT applications have different delay requirements, most are generally delay-tolerant [28]. As such, it makes sense to *bundle* multiple message transfer requests together and grant them later at once instead of granting individual requests immediately upon their arrivals, specially in scenarios where an IoT device aggregates sensor readings from multiple sensors.

As mentioned above, the side effect of bundling is the increased delay experienced by applications. Thus, there is a need for a flexible bundling algorithm that allows applications to trade increased delay for reduced energy consumption. While our solution for the previous problem (request bundling on smartphones [35]) serves that goal, it falls short in capturing the effect of the DRX mechanism. Specifically, the On/Off model used before, is a reasonable approximation model for radio energy consumption on smartphones. However, the On/Off model does not capture the effect of DRX, which would lead to significant overestimation or underestimation of the radio energy consumption, and hence, the inefficient management of the IoT device energy. Thus, due to the constrained energy resources in IoT devices, an accurate modeling of the DRX mechanism is required.

To this end, we study message bundling algorithms that are tailored to the specific operation of LTE radios and the DRX mechanism. We consider a cost minimization problem

that has the same objective defined for the previous problem, *i.e.*, a weighted summation of energy and delay costs. However, energy cost is modeled using a three state model, which is based on the behavior of the LTE radio. The weight factor in the objective function can be used to balance the energy-delay tradeoff based on the IoT traffic type (delay tolerant or delay sensitive) and also power constraints of the IoT device.

1.5 Contributions

- First, we study the problem of *latency and energy reduction for mobile web browsing*. In order to eliminate the initial round-trip time required to fetch the base HTML file of a page, we propose the design and implementation of a system called WebPro. WebPro relies on a network proxy that builds an up-to-date database of resource lists for the websites visited frequently by network users. The proxy resides in the wired part of the network, and hence can afford to pro-actively build and refresh the resource list database periodically. When a request for a webpage comes to the proxy, it simultaneously fetches the base HTML and all referenced objects required to render the webpage using the corresponding resource list stored in the local database. Once all the required objects of a webpage are fetched, the proxy packs them in a bundle and sends it to the mobile device. By implementing bundling, WebPro reduces radio energy consumption as well since it eliminates unnecessary state promotions and demotions in mobile's radio for each of the small objects.

  We have built a working prototype of WebPro and have conducted live experiments over WiFi and LTE networks. Our results show an average of 26% reduction in page load time for a mix of popular web sites chosen from categories such as news, sports and shopping. Moreover, in comparison to another well-known proxy-based solution, WebPro provides delay reductions ranging from 5% to 51% for a variety of web sites.

- After the first problem, we study *balancing energy-delay tradeoff for request bundling on*

smartphones. To minimize the bundling cost, we design a deterministic online algorithm called the *Break-Even* algorithm (BE), which is inspired by the classic Ski Rental problem [36]. Specifically, BE does not automatically grant each data transfer request upon its arrival. Rather, it bundles them together and makes a single grant when the energy cost and weighted delay cost associated with that grant become equal. Our algorithm is considerably general and can accommodate different definitions of the delay cost. For two commonly used delay functions *i.e.*, *cumulative* and *max* delay functions, we perform a detailed competitive analysis of BE with respect to the optimal offline algorithm that knows the future data transfer requests in advance. We prove that, 1) If the delay cost is defined as the cumulative delay of individual requests in a bundle, then BE achieves a competitive ratio of 1 or 4, depending on the value of the weight factor. 2) If the delay cost is defined as the maximum delay of individual requests in a bundle, then BE achieves a competitive ratio of 1 or 2, depending on the value of the weight factor. 3) In both cases, the competitive ratios, *i.e.*, 4 and 2, are tight.

We evaluate the performance of the proposed algorithm and the accuracy of our results in a range of realistic scenarios using both model-driven simulations and real experiments on a smartphone. Our results show that depending on the delay tolerance level of a user, energy savings ranging from zero (delay intolerant) to about 100% (delay tolerant) can be achieved using our algorithm.

- Finally, we investigate the problem of *online energy management in IoT applications*. To minimize the message bundling cost, we use the same online policy of the Break-Even algorithm introduced above (*i.e.*, make a grant when the energy cost and weighted delay cost associated with that grant become equal). However, the detailed implementation of the BE is adjusted to account for the new energy model that is based on the behavior of the LTE and DRX mechanism. Our detailed analysis shows that, depending on DRX and application parameters, our algorithm is 1, 2, or 4-competitive with respect to the optimal

offline algorithm that knows the entire sequence of application messages a priori.

To assess the performance of BE, we conducted an extensive set of model-driven simulations under a wide variety of realistic conditions. We also collected real traces from an experimental IoT testbed and used them on an Android-based LTE smartphone to empirically evaluate our online algorithm. Our results show that in most realistic scenarios, BE achieves an empirical competitive ratio less than 2. Also, depending on application requirements, energy savings ranging from zero to about 100% can be achieved using our algorithm. We also observe that DRX has a significant effect on energy consumption, which is neglected by the existing On/Off models.

1.6 Published work

The results of the research presented in this thesis were published in the proceedings of three conferences and a journal article. Specifically, the results of the research on latency and energy reduction for mobile web browsing appeared in IEEE/ACM IWQoS 2015 conference [37] and Elsevier Computer Networks journal [38]. Most of the results on energy-delay tradeoff for request bundling were published in the proceedings of IEEE INFOCOM 2017 conference [35]. Finally, the results of our study on energy management in IoT applications will be published in the proceedings of IEEE INFOCOM 2018 conference [39].

1.7 Thesis organization

The rest of the thesis is organized as follows. Chapter 2 provides the required background information for discussions in this thesis. It then presents a review of the related work for the problems considered in this thesis. Chapter 3 studies the problem of latency and energy reduction for mobile web browsing. Chapter 4 studies the problem of energy-delay tradeoff for request bundling on smartphones. Chapter 5 studies the problem of online

energy management in IoT applications. Finally, Chapter 6 concludes the thesis and presents directions for future research.

Chapter 2

Background and Related Work

2.1 Background

2.1.1 HTTP protocol

The HyperText Transfer Protocol (HTTP) is an application-layer protocol being widely used by mobile applications. As a request-response protocol running on top of TCP, it defines the structure of messages exchanged between a client program (e.g., a browser) and a server program (e.g., a web server). One of the most common use cases of HTTP is to retrieve a webpage from a web server. A typical webpage consists of a base HTML file and several embedded objects that are referenced in the base HTML file with their Unified Resource Locators (URLs). In the simplest form, opening a webpage involves downloading the HTML file, parsing it to identify the referenced objects and then fetching those objects.

Similar to other protocols, HTTP is always evolving. Until 1997, browsers and web servers supported HTTP/1.0. Since the standardization of HTTP/1.1 in 1999, it has been the dominant protocol implemented by browsers and web servers. The next version of HTTP, HTTP/2.0, was standardized in 2015 [40].

Persistent and Non-Persistent Connections. There are two approaches for requesting web objects using HTTP. The first approach is called HTTP non-persistent connection and requires each object to be transferred over a separate TCP connection. This way, after sending the response, the web server closes the TCP connection. As a result, the connection gets terminated by the time that the client receives the response object. While simple from the implementation perspective, this approach increases the load on HTTP servers as they

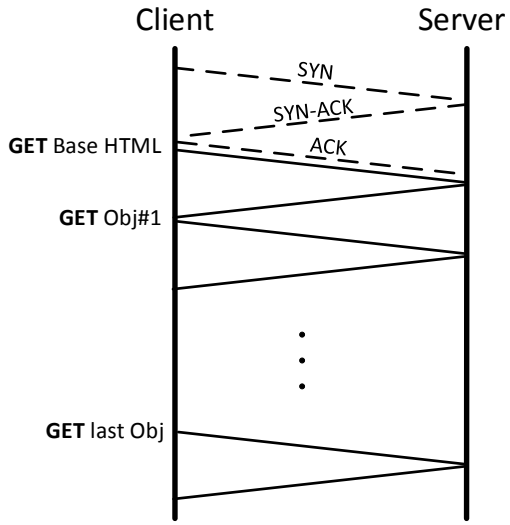


Figure 2.1: HTTP Persistent Connection [1]

should maintain TCP buffers and variables for every individual TCP connection that is established for each object [41]. Also in this approach, fetching each object incurs a delay of at least two round-trip-times (RTTs), one for establishing the TCP connection and one for fetching the object from the server.

The other approach is called HTTP persistent connection and was introduced in order to improve web page loading performance. In this approach, the TCP connection is kept open after sending a response by the server and the subsequent HTTP transactions reuse the same connection. This enables the client to use one connection for downloading all the objects of a page or even for downloading all the webpages residing on the same server. The HTTP server will close the TCP connection after a certain period of inactivity on that connection. Notice that HTTP clients and servers use persistent connections by default [41]. Figure 2.1 shows how a persistent connection can be used to load a webpage. Notice that after completing the first two parts of the three-way handshake, it takes one RTT to receive the base HTML file and then fetching each referenced object will incur another RTT.

Pipelining. HTTP pipelining is one of the optimization techniques introduced in HTTP 1.1 and it enables the client to send multiple HTTP requests back-to-back without waiting

for replies to previous requests. Pipelining is usually suggested as a way to dramatically reduce page load times over high latency connections. However, HTTP pipelining suffers from the Head of Line (HOL) blocking problem. The reason is that the protocol requires the responses to requests to be received in the same order that they were requested. As a result, a slow-to-generate resource in the server can block fetching subsequent objects [42].

Concurrent TCP Connections. Given that most modern webpages contain 10s or 100s of small objects, current browsers try to accelerate the page load process by using multiple TCP connections in parallel. This way, they can also circumvent the low initial congestion window size of TCP's slow start phase and attain a better utilization of the link's bandwidth [42]. In particular, modern browsers such as Chrome and Firefox use six concurrent TCP connections per host. Usually there is also an upper limit on the total number of concurrent connections across all domains [1] as the benefits of concurrent connections diminish by increasing the concurrency.

2.1.2 Browser anatomy

Here we briefly introduce the process of loading a webpage in a typical browser. This process starts with a user requesting a webpage through the browser's user interface. This will cause the *object loader* component to download the base HTML file of the page. After receiving the base HTML file, it is fed into an *HTML parser* that parses the page iteratively and constructs the intermediate representation of the page, called *Document Object Model (DOM)* tree. Parsing the base HTML reveals the set of embedded objects (also called subresources [43]) in a page. These objects could be of different MIME types including HTML (e.g., IFrame), JavaScript, CSS, image and Media [26]. The object loader will fetch these objects for further processing. It may have to fetch them from web servers over the Internet or from the browser's local cache. Of those subresources, JavaScript and CSS files need to be evaluated. Hence, there is an evaluator component in the web browser that

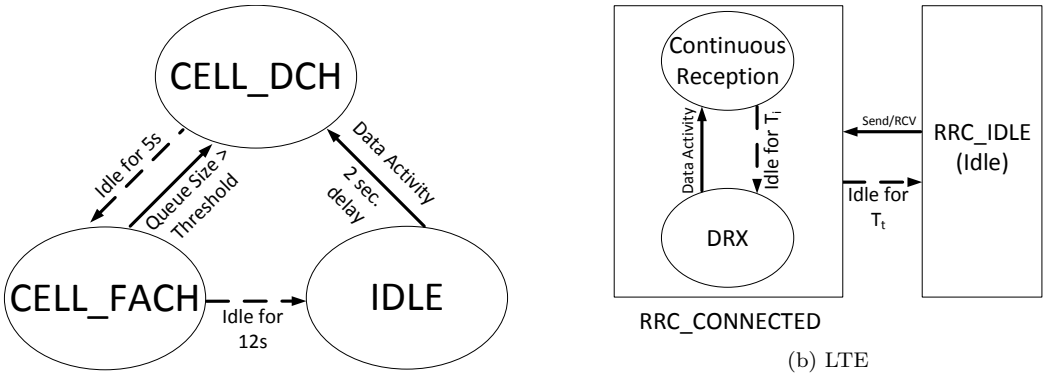
evaluates these objects and manipulates the DOM tree accordingly. Javascript files can add dynamic content to webpages and CSS files define the attributes of visible webpage elements such as font and colour. Notice that evaluating JavaScript or CSS files can result in new object requests [44]. Finally, a browser's rendering engine is responsible for visualizing the page using the constructed DOM tree. Rendering engine consists of a layout component which computes the size and screen locations of elements [26]. It also includes a painting component that generates the final representation of the page by putting the actual pixels on the screen [45].

2.1.3 Energy consumption of cellular radio

Recent years have witnessed an increase in the popularity of mobile devices such as smartphones and tablets. These devices are equipped with radio interfaces that enable communication over wide-area cellular networks. As a result, mobile devices have become one of the major choices for accessing popular network applications such as web browsing and email. However, wireless radio controller of a mobile device has a high energy consumption and can quickly drain the battery of the device.

Specifically, in cellular networks there is a state machine associated with each device that controls its energy consumption and allocated channels. Even though 3G and LTE networks share some general principles in their state machines, there are some differences in their behaviour. The general idea is to carefully manage the battery power of the device and radio resources of the network by putting the device in specific states.

3G State Machine. Figure 2.2(a) depicts the state machine employed in 3G networks. In IDLE state there is no radio resource allocated to the device and the user equipment (UE) does not consume any power in this state. When base station detects any traffic to/from the device it promotes the device to the *CELL_DCH* state by allocating a dedicated channel to UE. This state consumes significant amount of battery power (about 800/600 mW depending



(a) 3G

(b) LTE

Figure 2.2: RRC state machines for the 3G/LTE networks

on the carrier [46]). The promotion from IDLE to DCH takes approximately 2 seconds [46]. In order to be energy efficient, the device can't stay in the DCH state all the time. So if there is no traffic activity for ~ 5 seconds, the device will demote to the *CELL_FACH* state with the operating power of around 460 mW. Once in the FACH state, the user device can send/receive data through a shared channel at a lower rate. Notice that the use of the FACH state for 3G networks is optional and is not supported across all carriers [47]. If the device remains inactive for another ~ 12 seconds it will go back to the IDLE state.

LTE State Machine. As shown in Figure 2.2(b), energy consumption in LTE networks is characterized by a state machine consisting of two states, RRC_IDLE and RRC_CONNECTED. RRC_IDLE (or idle state) represents the lowest energy state and data transmission to/from the UE in this state will cause a promotion to the RRC_CONNECTED state in about 200 ms. Once promoted, the user equipment enters the Continuous Reception mode (also called *active* state) and consumes high power as it continuously monitors the physical downlink control channel (PDCCH) for scheduling information. The UE also starts an *inactivity timer*, T_i, in this mode which gets restarted every time a data transfer request is scheduled before the timer is expired. Otherwise, the expiry of the timer moves the UE to Discontinuous Reception (DRX) mode. In DRX mode, the UE periodically wakes up to monitor PDCCH only for short intervals (referred to as ON durations) and then goes to sleep at other times.

As a result, power consumption in DRX mode is higher than the idle state but is lower than the active state. Along with T_i, the UE also starts a timer called RRC *tail timer*, T_t, every time it enters the RRC_CONNECTED state. When there is no network activity for the duration of the tail timer, the UE moves to the idle state. Finally, it is worth noting that the promotion delays are shorter in LTE compared to 3G networks [9].

Each of the aforementioned state transitions consumes energy and also introduces some signalling overhead in the system. Also promotion delays and timers associated with each state can have different values across vendors and carriers.

2.1.4 Online algorithms

The basic assumption in the design and analysis of traditional algorithms is that an algorithm uses its complete knowledge of the entire input in order to generate output. However, this assumption does not necessarily hold in all practical applications. Some algorithmic problems in practice are *online* meaning that the input is revealed in parts and hence is only partially available at any time. Thus, an online algorithm is required to generate output without knowledge of the entire input. In other words, the online algorithm must make a decision every time a new input arrives. Also at any point in time, it is impossible to revoke the previous decisions made by the online algorithm [48].

Formally, assume that a sequence of requests $\mathcal{A} = \langle a_1, \ldots, a_m \rangle$ is presented to an online algorithm ALGO over time. These requests must be served in the order of their occurrence. Also when algorithm ALGO serves a_i, it has no knowledge about any request a_j with $j > i$. There is a cost associated with serving each request and the objective is to minimize the total cost incurred due to serving the entire request sequence [49].

A standard mechanism to evaluate the performance of online algorithms is *competitive analysis*. In competitive analysis, an online algorithm ALGO is compared to an *optimal offline algorithm*, which is an unrealistic algorithm that has full knowledge of the entire input sequence in advance and hence can serve it with minimum cost. For an input sequence

$\mathcal{A}$, let OPT($\mathcal{A}$) denote the cost incurred by the optimal algorithm and let ALGO($\mathcal{A}$) denote the cost incurred by the online algorithm ALGO. Algorithm ALGO is said to attain a competitive ratio of c (*i.e.*, be c-competitive), if for all possible input sequences $\mathcal{A}$, there exists a constant a that satisfies the relation ALGO($\mathcal{A}$) $\leq c$.OPT($\mathcal{A}$) $+ a$. Notice that the competitive ratio is independent of the input sequence and is always at least 1. Also the smaller the competitive ratio is, the better the online algorithm performs [50].

2.2 Related Work

2.2.1 Reducing Latency and Energy of Mobile Web Browsing

There is a large body of work on improving the performance, energy usage and wireless data consumption of web browsing on mobile devices. Here, we classify the work that is most relevant to ours.

Client-based Solutions. Traditional solutions based on client-side caching and prefetching fall in this category. As an example, the authors of [51] used a machine learning approach to model the web browsing signature for each individual user. This model can predict the future web access patterns, enabling a prefetching scheme to download web content before the actual user request.

Recently, there have been measurement studies to assess the effectiveness of client-side caching and prefetching in improving the performance of mobile web browsing. For instance, Ma *et al.* [52] conducted comprehensive measurements to characterize the performance of mobile web caching. They identified *redundant transfers* and *miscached resources* (providing out-of-date resources from cache) as the two main problems that negatively affect mobile web caching performance. Their investigations revealed *Same Content* (same resources having different URLs at different times), *Heuristic Expiration* and *Conservative Expiration* as the root causes of unsatisfactory cache performance. Wang *et al.* [43] used a web dataset collected from 24 iPhone users over a year to quantitatively evaluate client-only caching and

prefetching. Their results indicate that there is a limited efficiency gain due to caching and prefetching when it comes to mobile web browsing. Consequently, they proposed a new technique called speculative loading which predicts and loads the required resources of a page in parallel with the base HTML file of the page. However, their approach requires changing the mobile browser extensively, which limits its practical feasibility.

One major drawback of the client-only solutions is that any incorrect prediction can lead to downloading data that the user may never use. While not a significant problem in wired networks, this can waste the scarce resources of mobile battery and wireless bandwidth and hence harm user's experience rather than improving it in wireless networks. In order to accurately balance costs and benefits of prefetching, authors of [53] proposed a system level solution that provides explicit prefetching support to mobile applications. However, their solution requires extensive modifications of the existing applications. Another drawback of client-only solutions is that they cannot observe the aggregate behaviour of users and benefit from their common browsing activities which is at the heart of traditional caching techniques.

Protocol-based Solutions. SPDY by Google [54] is a new application layer protocol primarily designed for reducing latency of web browsing. SPDY multiplexes multiple data streams over a single TCP connection. It also enables unsolicited push of embedded objects by web servers which can speed up the resource loading process in the browser. Combined with other advanced features, SPDY can be very effective in reducing the web browsing delay [54]. However this protocol relies on web server support and given that only 8.9% of all websites support SPDY [55], its impact so far has been rather limited. Also the next generation HTTP protocol, HTTP/2, which was standardized on February 2015, evolved from SPDY [56] and therefore inherits most of its features. However, there are a few differences between SPDY and HTTP/2.0 for example in their header compression algorithms [40].

With server push being one of the main novelties in SPDY/HTTP 2.0, there have been

several proposals to resolve some of its limitations or to further improve the efficiency of this feature. For example, notice that the server is oblivious of client's cache status and by pushing content that already exists in client's cache, it can waste bandwidth and battery of the mobile device. To address this issue, Khalid *et al.* [57] proposed sending cache hints from client to server in the form of bloom filters. To further adjust the performance of server push, they also proposed the ideas of *half-push* and *half-pull*. In half-push, the server pushes the content to an edge proxy rather than the client and in half-pull the client requests the content to be brought to the proxy without traversing the last mile. Finally, authors of [23] proposed a novel framework that uses the server push feature in HTTP/2 to preemptively push resource lists of the requested page and all its subpages, to the client. By using these cached meta files, a future request for a subpage can be issued in parallel with the subresource fetching requests of that page. In this scheme, client incurs little extra bandwidth overhead due to meta data transfers but can benefit from speedup in downloading subpages.

Infrastructure-based Solutions. Some of the previous work in this category has tried to improve the energy efficiency of mobile web browsing. Aggrawal *et al.* [58] proposed a cloud-based proxy system to reduce the energy consumption of the smartphone's data communication by employing aggregation, redundancy elimination and opportunistic scheduling when downloading web objects from the network. Wang *et al.* [18, 59] presented a dual-proxy architecture called EEP that utilizes bundling and compression to reduce the energy consumption of web browsing in 3G/WLAN networks.

There are also studies that try to reduce both power consumption and delay of mobile web browsing. For example, Zhao *et al.* [60] proposed a Virtual-Machine based architecture in which a VM-hosted proxy performs all the computation expensive tasks of mobile browsing and sends a screen copy of the rendered page to the smartphone. However, as mentioned in [45], offloading compute-intensive operations when loading a webpage has neg-

ligible benefits compared to the improvements resulting from resource loading optimizations. Also Sivakumar *et al.* [19] proposed PARCEL which uses the same architecture as in EEP while providing the proxy with the flexibility to optimize the bundle size in a cellular friendly manner.

Finally, this category includes studies with the goal of reducing latency of mobile web browsing. Some of them achieve this goal by reducing the amount of data transmitted because of web browsing [61, 62], while others employ solutions that directly deal with network access delay [63]. For example, Opera Mini [61] and Amazon Silk [62] are cloud-based browsers that offload portions of the page load process to cloud-based proxies. These browsers are widely used today based on the common belief that using compression proxies reduces data usage of mobile web and thus, reduces latency. However, the results of a recent measurement study in [64] reveals that using compression proxies in good network conditions can increase page load time rather than improving it. As a result, they design and implement a framework called *FlexiWeb* that decides whether to fetch a resource from middle box or the original server based on the object size and the network conditions. In line with the findings of [64], a recent study by Sivakumar *et al.* [65] also shows that cloud-based browsers are not always superior in terms of responsiveness and energy consumption, especially in dealing with client interactions.

Instead of directly reducing page load time in mobile web browsing, authors of [10] proposed *KLOTSKI* that aims at improving mobile user's quality of experience by delivering as many high utility resources as possible within tolerance limits of mobile users (3-5 seconds). To this end, KLOTSKI employs a cloud-based proxy to capture and update different properties of websites such as dependency structure, resource sizes and positions on rendered displays and stores them in the form of a compact fingerprint. When loading a page, those fingerprints are used to dynamically reprioritize delivery of different resources.

The closest work to ours is EEP by Wang *et al.* [18, 59] and PARCEL by Sivaku-

mar *et al.* [19]. While their focus is on reducing energy consumption by batching and compression [18, 19, 59], our main goal is latency reduction using the speculative loading technique. These solutions are orthogonal to each other and can be used in combination to create a solution that is both energy efficient and low latency.

2.2.2 Energy-Delay Tradeoff for Request Bundling

We divide the prior work related to our work into the following categories.

2.2.2.1 Tail Energy Mitigation

There are three major classes of solutions devised to mitigate the effect of tail time. First class consists of solutions that propose optimization and dynamic reconfiguration of the tail time in response to the changing traffic pattern of smartphones [66, 67]. The fact that configuring RRC parameters is controlled by cellular network carriers is a hurdle in practical implementation of these solutions. The second class consists of solutions that utilize the fast dormancy feature [68, 69]. In these solutions, the user equipment (UE) uses detailed application-specific knowledge to predict the end of a network usage period and proactively request demotion of radio to a low power state. However, the high dynamics in UE traffic, can lead to low prediction accuracy and hence eliminate the benefits of fast dormancy. Finally, there are solutions based on bundling (discussed in the next subsection) that delay data transfer requests and transmit them in bundles to reduce the tail energy.

2.2.2.2 Traffic Bundling

The following is a brief review of several works on bundling that are more relevant to our research.

Packet Bundling. Packet bundling can be implemented at the link layer irrespective of the applications running on the device. Deng *et al.* [12] proposed a learning algorithm to predict the traffic pattern in order to decide when a packet bundle should be granted. In another work, Dogar *et al.* [70] showed that the bandwidth discrepancy between wired and wireless

segments of a connection can create small idle gaps between successive packets. They then designed a proxy-based system that bundles multiple packets together in order to maximize the idle time between bundle grants. Alonso *et al.* [20] proposed bundling packets in order to improve the energy efficiency of the DRX mechanism in LTE networks. Assuming a Poisson packet arrival process, they computed the optimal size of bundles that minimizes the radio energy consumption, while ensuring a bounded average packet delay.

Mobile Web Browsing. Hoque *et al.* [71] showed that when viewing web pages, there are idle gaps between consecutive web object downloads. This is due to the fact that inter-object dependencies in web pages require some processing, for example to evaluate Java scripts, which can create delays between object downloads [26]. To reduce the radio energy consumption of mobile web browsing, a proxy-based architecture was proposed in [18,19], in which a network-hosted proxy fetches web objects from remote servers and then sends them to the client device in bundles. In particular, using the heuristic assumption of equal-sized bundles, Sivakumar *et al.* [19] computed the optimal number of bundles that minimizes the radio energy usage of visiting a web page.

Delayed Data Offloading. While not directly related to minimizing the radio on time on smartphones, the problem in delayed data offloading has a similar structure to the bundling problem considered in this thesis, and hence any solution for the bundling problem will be of benefit to the data offloading problem as well. In delayed data offloading, the problem is to decide how long to wait until a WiFi network becomes available to offload data transfer requests from the cellular network to the WiFi network [72, 73].

Delay-Tolerant Applications. Balasubramanian *et al.* [13] proposed a threshold-based bundling algorithm called TailEnder which achieves a competitive ratio of 2. The idea is that data transfer requests from delay-tolerant applications can be postponed up to a deadline without any penalty. The TailEnder requires modification of applications so that they can inform the bundling algorithm of their deadlines.

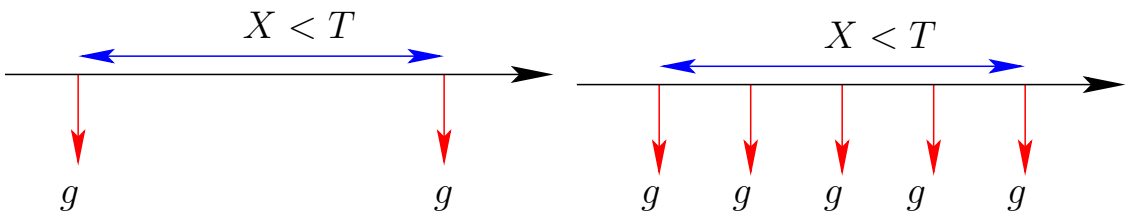
Mobile Code Offloading. In practice, it is difficult to know the deadlines for data transfer requests particularly for non-delay-tolerant applications. Instead, Xiang *et al.* [16] formulated the bundling problem in the context of mobile code offloading as a cost minimization problem, where the cost of bundling is given as a function of both energy and delay. Thus, a bundling algorithm may delay bundles arbitrarily in order to reduce its energy cost as long as it is willing to accept the increased delay cost. Then, based on [74], they devised an online algorithm to minimize the cost of bundling.

With the exception of [16], the aforementioned works on bundling do not have the flexibility to achieve different energy-delay tradeoffs. Indeed, the problem setting considered in our work is similar to the one considered in [16]. There are, however, significant differences between the two works, as discussed next in section 2.2.2.4.

2.2.2.3 Ski Rental Problem

Ski rental problem is about a skier who wants to go skiing and must decide whether to buy skis or to rent them. Assume that it costs \$1 to rent skis per day and it costs \$$B$ to buy them. With foresight, it is better to rent the skis, if the number of skiing days is less than B, and it is better to buy them right away if the skier plans to go skiing for more than B days. However, the challenge is that the skier does not know, ahead of time, the number of days s/he will be skiing. The number of skiing days is unknown due to various reasons such as unpredictable weather conditions, or possible loss of interest. So, every day when the skier goes skiing, s/he has to make an online decision, on whether to rent skis or to buy them.

There is a simple deterministic online algorithm for the ski rental problem, called the Break-Even algorithm. The Break-Even algorithm dictates the skier to continue renting the skis up until the B^{th} day, at which point the skier should buy the skis. It has been shown that the competitive ratio of the Break-Even algorithm is 2 [36]. There is also a randomized online algorithm that achieves an expected competitive ratio of $e/(e-1)$ [75]. It has been shown that the best achievable competitive ratios for the ski rental problem by



(a) Two grants. (b) Multiple grants.

Figure 2.3: Energy cost is variable: Scenarios (a) and (b) have the same energy cost even though they have different number of grants.

any deterministic and randomized algorithm are 2 and $e/(e-1)$, respectively [75].

2.2.2.4 Dynamic TCP Acknowledgement

One would appreciate the resemblance between our problem (hereafter called *Energy-Aware Bundling* or EnerB) and the *dynamic TCP acknowledgment* problem (DynAck) [76]. In DynAck, there exists the possibility to acknowledge multiple packets simultaneously by bundling their individual acks together and sending a single cumulative acknowledgement. The goal is to select the timing of cumulative acks in a way that minimizes the acknowledgement cost plus the delay cost incurred due to delaying acks. DynAck is a generalization of the ski-rental problem and can be solved by a 2-competitive deterministic algorithm and a randomized $e/(e-1)$-competitive algorithm [74]. The problem can then be cast as a variation of EnerB, in which the energy cost is replaced by the acknowledgment cost, *i.e.*, the number of acks sent. Indeed, this relation was exploited in [16] to design a bundling algorithm for mobile code offloading. There is, however, a subtle difference between problems EnerB and DynAck, which completely changes the problem. *Whereas the cost of sending an ack is constant, the energy cost of making a grant is variable.* Specifically, the cost of sending an ack is always 1 (no matter how long the algorithm waits to send the ack), and thus, the acknowledgment cost incurred by the algorithm is simply given by the total number of acks sent.

In contrast, in EnerB, the energy cost of making a grant is *variable* as it represents the amount of time the radio has been on. For a bundling algorithm A, let E_A denote its energy

cost. In this case, the energy cost E_A is the summation of radio on times over the duration of the algorithm, and hence the number of grants may not have any relation to E_A. Consider the scenarios depicted in Fig. 2.3, where T denotes the tail time. Grants are specified by g on the figure. In Fig. 2.3(a), there is one grant at the beginning and one grant at the end of an interval of length X. In Fig. 2.3(b), in addition to the grants at the beginning and end of the interval of length X, there are three other grants during the interval. Since $X < T$, in both scenarios, the radio is on for the entire duration X. Thus, the energy cost of the bundling algorithm in both scenarios is equal to X, even though there are different number of grants during the same interval. Thus, as opposed to the analysis presented in [16, 74, 76], if algorithm A makes more grants than algorithm B, we cannot conclude that $E_A > E_B$, as shown in Fig. 2.3. This relation is at the heart of the analysis presented for the performance of the online algorithms proposed in [16, 74, 76]. As shown later, we design a completely new approach to analyze the performance of our algorithm as the approach taken in the above mentioned works does not apply.

2.2.3 Energy Management in IoT Applications

There is a large body of work on DRX-aware radio energy management schemes. The most relevant categories related to our work include:

2.2.3.1 DRX optimization

There have been several studies on improving the energy efficiency of IoT devices by optimal configuration of DRX parameters (*e.g.*, see [77, 78], and references therein). The common approach in these works is to model the effect of DRX parameters on energy and delay at the UE side, and then determine the optimal DRX parameters that achieve a desired tradeoff between energy and delay. To model energy and delay, typically modeling assumptions are made about the traffic arrival process and other aspects of the system. For example, [77] developed a Markov model to characterize DRX effect on energy and delay assuming a Poisson

traffic arrival. In contrast, our analysis is independent of any specific assumption about the incoming traffic. Also, the problem considered in this thesis is orthogonal (and complementary) to the existing work on DRX. In particular, we design our algorithm assuming that the DRX parameters are already configured and set using one of the above optimization models.

2.2.3.2 DRX enhancement

There have also been proposals for modified versions of DRX that address the energy efficiency requirement of IoT devices. For example, the authors of [79] proposed to enhance the DRX mechanism with a quick sleep indication, where the base station (called evolved NodeB or eNB) can inform the device to go to sleep when there is no incoming traffic. Also in order to increase the time spent in DRX mode while achieving a bounded average packet delay, [20] proposed a packet coalescing mechanism, where the eNB delays transmitting packets to the UEs in DRX mode until their downstream queues reach a tunable threshold. However, these schemes require changes to the operation of current eNBs which could hinder their deployment and adoption.

2.2.3.3 DRX-aware scheduling

Several recent works have proposed scheduling strategies that take DRX operations into account. For example, Liang *et al.* [80] suggested using a DRX selection algorithm along with a cooperating DRX-aware scheduling algorithm at eNB in order to satisfy QoS requirements of IoT applications. However, most IoT traffic is uplink, and hence their algorithm is not sufficient for most IoT applications. Wang *et al.* [29] proposed an IoT device-based uplink scheduler that balances device power consumption and the network signaling load. Their design choice requires end devices cooperating in signal load reduction of the network.

2.2.3.4 Smartphone request bundling

As discussed earlier in subsection 2.2.2.2, there are some existing works that address the energy-delay tradeoff associated with bundling [13, 16]. However, these works along with our

work in Chapter 4 consider an On/Off radio model which does not capture the operation of the DRX mechanism in LTE networks.

Chapter 3

Reducing Latency and Energy of Mobile Web Browsing

Recent advances in cellular technology have given rise to the widespread adoption of mobile devices such as smartphones and tablets. Among numerous mobile apps, web browsing is still one of the most popular applications on mobile devices. Due to limited bandwidth and longer access delays in wireless networks (more specifically, cellular networks), however, web browsing is generally slower on mobile devices, which could frustrate users and lead to lost online business opportunities. For example, it is estimated that a 2 second increase in the load time of Bing's home page can reduce revenue per user by 4.3% [81].

Prior work [45, 82] has shown that different from desktop computers, there is a new set of factors causing the slow browsing experience on smartphones, which calls for solutions tailored to mobile web browsing. Some of these factors are:

1. Compared to the enterprise Ethernet typically used by desktop computers, wireless hop has longer access delays which dominate the end-to-end round trip time (RTT) and consequently result in longer RTTs. The long network RTT makes resource loading the bottleneck of web browsing on smartphones. On the contrary, compute intensive operations such as scripting, style formatting and layout are the bottleneck in desktop browsers.

2. Limited processing power of smartphones affects the resource loading process as it is associated with network stack and OS services.

3. Many webpages are not designed specifically for web browsing on mobile devices. For example, analysis of the traces of 25 iPhone users in [45] shows that over half of the webpages visited by smartphone users are not optimized for mobile devices or are non-mobile webpages.

Recently, there has been a significant amount of work on reducing the latency of mobile web browsing [1, 23, 43, 51, 53, 54, 57, 60]. Some of these efforts rely on modifying the web access protocol. For example, SPDY [54], a new protocol designed by Google, aims to minimize the latency of web browsing by adding request multiplexing, support for prioritization and a number of other advanced features. However, this solution requires changing the client and server side software which limits its widespread adoption. There are also prior attempts that rely on client side optimizations. This category includes solutions based on client side caching [21] and prefetching [51, 53] along with a recently proposed technique called speculative loading [43]. The short expiration times of most web objects limit the efficiency of caching techniques, while prefetching solutions suffer from wasted wireless bandwidth and battery resources that result from incorrect predictions (not a problem on wired desktop browsing). On the other hand, speculative loading technique relies on extensive changes to the mobile browser which is a hurdle to its adoption.

Other noticeable solutions are those based on network proxies. These solutions mostly try to reduce the computation time or energy consumption of web browsing by delegating some tasks involved in opening a page to a powerful entity in the network such as a cloud-based proxy [60, 61, 83]. One of the major advantages of employing a network-based proxy solution is that a proxy can offer a better improvement by learning and exploiting the aggregate browsing behaviour of a diverse mix of mobile users which is not possible in client-only solutions.

Specifically, some network-based solutions such as VMP [60] and Opera Mini [61] aim at offloading compute-intensive operations of the page loading process to a proxy. However, it has been shown that optimizing compute-intensive operations leads to only marginal improvements in the overall page load performance [45]. Thus, other solutions such as EEP [18, 59] and PARCEL [19] try to offload *resource loading* operations to a network-based proxy in order to improve page load performance. Specifically, in these solutions, proxy

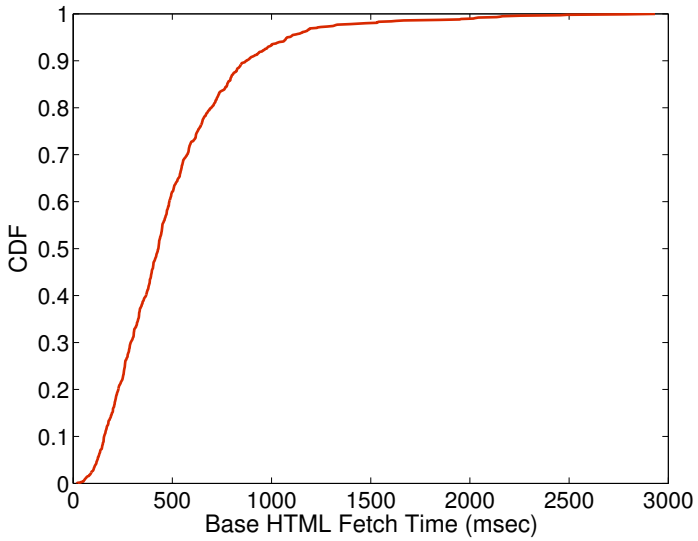


Figure 3.1: CDF of the time to fetch the base HTML file for Canada's top 100 websites. In the median case, it takes 430 ms to download the base HTML file. However, this time can go beyond 1 second in some cases.

retrieves the base HTML file of the page and parses it to discover referenced objects, which could be then fetched and transmitted to the client in a bundle (in order to reduce energy consumption of the mobile device).

One essential aspect of such proxy-based solutions is that the proxy can build and transmit the bundle only after it has finished downloading all the embedded objects of a page. Considering the request-response nature of the HTTP protocol, discovering the list of the referenced objects requires at least one RTT in order to fetch and parse the base HTML file. Also, one or more redirections might be involved before arriving at the base HTML file which can further delay the realization of the web objects.

To gain a better insight, we measured the latency of downloading the base HTML file for the top 100 Canadian websites [84] from a desktop computer connected to campus Ethernet. Because of the redirections, this time might be different from the RTT between our device and the corresponding web server. Figure 3.1 shows the cumulative distribution function of the time to fetch the base HTML file of each site. In the median case, it takes 430 ms to fetch the base HTML file. However, over 6% of the cases experience latencies beyond 1 second.

Also according to the measurement results in [26], the base HTML fetch time constitutes the largest fraction of the network time for loading a page. *This implies that there is a potential for optimizing mobile browser performance by eliminating the initial fetch time.*

The rest of the chapter is organized as follows. Section 3.1 presents a high level overview of our approach in addressing the above issue. Section 3.2 introduces our proposed solution in detail and discusses different aspects of it. Section 3.3 offers results on the performance evaluation of the system.

3.1 Our Approach

In an effort to reduce the latency of mobile web browsing, we propose the design and implementation of a system that aims at eliminating the initial round-trip time required to fetch the base HTML file of a page. Our solution, called WebPro, is built on two cooperating proxies, one of which resides in the mobile device and the other one, remote proxy, is deployed inside the network, preferably as close to the user as possible (see Figure 3.3). When a user wants to visit a page, the remote proxy will fetch all the required objects on behalf of the mobile device. After downloading all the objects, the remote proxy packs them in a bundle and pushes it to the local proxy, which will serve all browser's requests locally. In this dual proxy architecture, not only we are able to significantly reduce page load time but also reduce energy consumption by implementing bundling to eliminate unnecessary power state promotions and demotions in mobile's radio for each of the small objects [18, 85].

In order to fetch all the required objects of a page, the remote proxy employs the *speculative loading* technique [43]. The main idea behind this approach is to bypass the extra time for fetching and parsing the base HTML file, by using a previously recorded list of all the required objects for a webpage, hereafter called the webpage *"resource list"*. Figure 3.2 presents the resource list for an example webpage. We observed that the amount of change in the structure of the webpages within a few hours is relatively low and hence it should

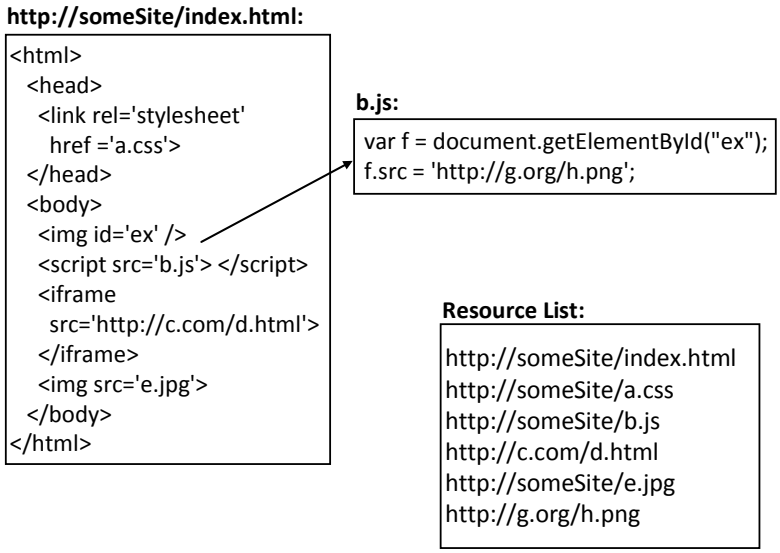


Figure 3.2: Resource list for an example webpage. This webpage contains a CSS, a JavaScript, two images and an HTML iframe. Notice that the embedded JavaScript file itself refers to another image file which can be identified only after the JavaScript file is fetched and processed.

be feasible for a proxy to keep track of such changes and maintain an updated resource list of the popular pages (pages that are popular among its users). Note that maintaining the resource lists of the webpages is different from caching the actual web objects, the majority of which can not be cached or have a short expiration time [43]. Nevertheless, such legacy caching and prefetching techniques can be added to our system if desired.

Maintaining an updated set of resource lists is achieved by enhancing the remote proxy with a *profiler* that periodically visits popular websites and records their resource lists in a metadata repository. Considering that the proxy resides in the wired part of the network, it can afford to pro-actively fetch webpages and construct their resource lists for the most popular websites in the network. Such a profiling module can be easily integrated with the operational activities of high-performance dedicated middle-boxes that are already deployed by most mobile operators for caching, traffic monitoring and optimization purposes [86].

This way, the first step in loading a page at the remote proxy will become checking the metadata repository. In the case the repository contains the resource list of the page, multiple parallel connections will be used to fetch all the objects of the page from possibly

different web servers. Otherwise, the remote proxy will employ a web engine to load the page by first fetching the base HTML file and then loading the discovered objects. WebPro's profiler employs a web engine to perform all the steps involved in loading a page except rendering. This way, profiler will be able to record all the requests that result from parsing as well as script evaluations. We also implemented a filtering module to prevent profiler from recording changing URLs that result from third party advertisements and tracking systems.

In order for the metadata repository to contain the resource lists of the majority of the requests, profiler should employ an effective mechanism to identify popular URLs in the network. Considering the flow of URLs into the proxy as a data stream, we exploit a well-known algorithm in data mining community, *space saving*, for identifying the popular URLs at the proxy. Our experiments using traffic traces collected from University of Calgary's Internet link show that maintaining a metadata repository for the top-1000 URLs in the network, allows a timely update of their resource lists by the profiler while providing a high hit ratio to the user requests.

Using resource lists at the proxy also enables WebPro to avoid going through the iterative process of exploring objects in a webpage. In other words, it enables WebPro to break the inter-object dependencies in a webpage. The most common form of such dependencies happens when an embedded object itself refers to another object, similar to the example in Figure 3.2. Accordingly, our experiments using carefully designed synthetic webpages reveal that the benefits of WebPro will extend as the number of dependencies in a webpage increases.

We emphasize that in contrast to client-based approaches (e.g., [43]), WebPro is transparent to the end-points and does not require any changes to the client's browser. As a proxy, it exploits the common browsing activity across a diverse set of mobile users and hence provides a faster browsing experience. Moreover, in WebPro, the penalty of downloading wrong and unusable objects (in terms of wireless bandwidth usage and mobile battery consumption) is

negligible compared to that of client-based approaches as it resides in the wired part of the network. Thus, it can afford to pro-actively update the resource lists, which is very costly to implement on wireless clients.

We have implemented WebPro on Linux and have conducted an extensive set of measurement experiments. We believe that the common approach taken by proxy-based solutions EEP [18, 59] and PARCEL [19] is the state of the art and one of the most complete proxy-based solutions for improving web browsing performance on mobile devices[1]. We call this approach PBB (Proxy Based Browsing) and use it as benchmark to evaluate the performance of WebPro. In comparison to PBB, our scheme achieves lower page load times. Specifically in the case of a workload consisting of the 20 popular webpages from different categories, our approach loads 73% of the pages in less than two seconds while under PBB, only 28% of the pages load in that time. To the best of our knowledge, this is the first work to use the speculative loading approach in a dual proxy architecture for improving mobile user experience.

3.2 WebPro: Proxy-Based Speculative Loading

3.2.1 System Architecture

In order to eliminate the initial fetch time at the remote proxy, we take advantage of the speculative loading approach. The basic idea of speculative loading is to use the previously recorded knowledge about the structure of a website during the page load process. Our system, called *WebPro*, is depicted in Figure 3.3. WebPro equips the remote proxy with a profiling module that pro-actively and periodically loads webpages from a set of top visited websites and records their resource lists in a metadata repository. The list of top websites can be inferred from the web browsing behaviour of the users of the system. As will be discussed later, the memory footprint of keeping resource lists is very low, which means

[1]The difference between EEP and PARCEL solutions was discussed in section 2.2.1 of this thesis.

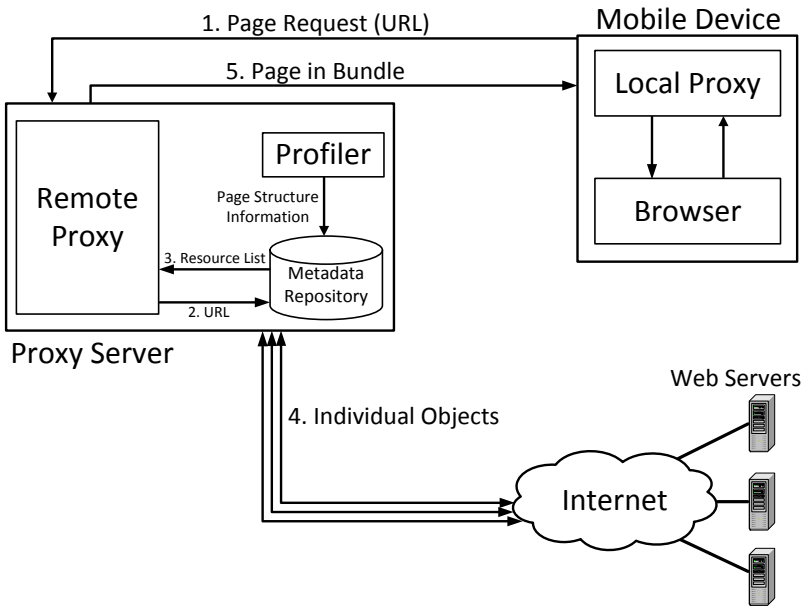
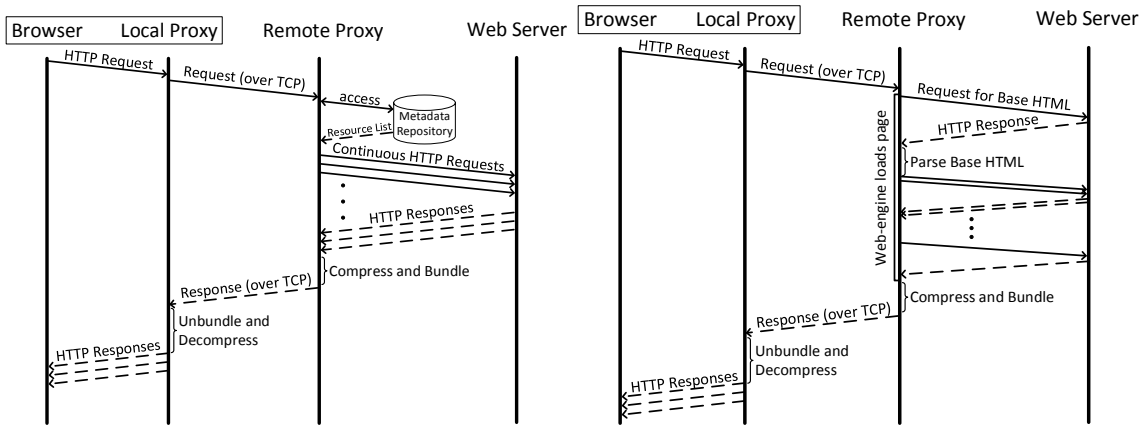


Figure 3.3: High Level Architecture of WebPro.

that the proxy can easily keep metadata for a large number (on the order of thousands) of websites.

After receiving a request to load a webpage at the remote proxy, if the resource list of that page already exists in the metadata repository, multiple parallel connections will be used to fetch the objects in the resource list. In case the remote proxy receives a request for the first time and notices the absence of the corresponding resource list, it will use the legacy approach of PBB by loading the page in a web engine. Once all the required objects of a webpage are fetched, the remote proxy packs them in a bundle and sends the bundle to the local proxy. Figure 3.4 shows the download pattern of WebPro and PBB. Notice that both WebPro and PBB bundle objects when transferring them from proxy to the client. The flow chart in Figure 3.5 shows the operations performed at WebPro's remote proxy for serving a user request.

A defining feature of WebPro is that the profiler on the remote proxy can always keep a fairly *recent* version of the resource lists for user requested webpages. However, the freshness of the maintained resource lists will depend on the frequency of change in the structure of



(a) WebPro (b) PBB

Figure 3.4: Downloading a Webpage with WebPro (a) and PBB (b).

the webpages. In Section 3.3.4.1, we will present measurement results indicating that on average the amount of such change within a few hours is relatively *small*. Therefore, given the abundance of the computation and communication resources at the remote proxy, it should be feasible for the profiler to capture the temporal changes in resource structures by updating its metadata repository in a timely manner. *Notice that doing so on the mobile device using a client-based approach is not feasible due to bandwidth and battery limitations.* Also it is noteworthy that an optimized implementation of the proxy will not penalize the page load times in the case of websites with rapidly changing structures (such as social media news feed sites), but it may not improve them either.

In order to learn and utilize the aggregate browsing activity of users in WebPro, whenever the remote proxy loads a page for the first time through the web engine, it also adds the corresponding resource list to the metadata repository. This way, the remote proxy will be able to exploit the common browsing activity across different users.

It is important to note the difference between WebPro and traditional proxy-based caching systems [87]. Those systems cache the actual content of web objects, which limits their efficiency as most web objects can not be cached or have a short expiration time [43]. However, with WebPro, the remote proxy just keeps a list of the referenced URLs and fetches a fresh copy of the corresponding objects at each page request. Despite this differ-

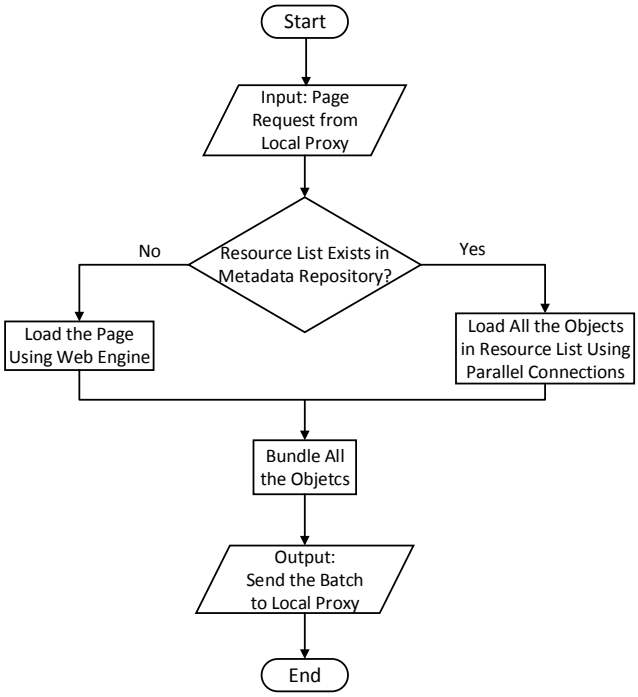


Figure 3.5: Flow Chart of Operations Performed at Remote Proxy

ence, WebPro could be augmented with traditional caching as well in case some objects are usable because there is plenty of storage/processing capacity available at the remote proxy.

3.2.2 Circumventing Webpage Dependencies

In addition to eliminating the initial HTML fetch time, there are other reasons that lead to a reduced page load time in our approach. Those reasons are based on the fact that the activities involved in the process of loading a page are inter-dependent and can block each other [26]. For example, some of the objects may be referenced by a JavaScript or CSS file and loading those objects depends on evaluating the referencing scripts. Also, downloading and evaluating a synchronous JavaScript file blocks HTML parsing during the page load process.

The immediate implication of such dependencies is that a web engine's resource loading operations are not fully parallel and discovering web objects can be further delayed because of script evaluations and other dependencies. However, WebPro can use a previously recorded

resource list and hence load all the required objects of a page without going through such dependent operations.

3.2.3 Identifying Popular Websites

As implied in previous sections, WebPro's profiler keeps a list of popular URLs in the network and by periodically loading those URLs, updates its metadata repository. It is clear that the benefits of WebPro will increase if the profiler maintains a URL list that achieves higher hit ratios for page requests in the network. The reason is that the presence of a user requested URL in the list of top URLs (hit occurrence) means that the proxy already has the updated resource list of that page in its metadata repository and hence can load the page faster. A simple approach for constructing such a list of URLs would be keeping a `SET` data structure and adding all the URLs to the `SET` upon their arrival at the proxy.

To gain a better insight, we applied this approach to network traffic traces that were collected from University of Calgary's Internet link. These traces contain summary information of all the HTTP transactions and are recorded by `Bro` [88], an open source Network Intrusion Detection System. We wrote an `AWK` script to extract the URLs of the landing pages from HTTP traces of six consecutive days (May 1, 2015 to May 6, 2015). We ran an offline analysis on the extracted traces and constructed the URL set by adding elements to it using the set union operation. Before adding a URL to the list, a dictionary lookup operation is performed to test whether it already exists in the list, and if so, a hit counter is incremented. In this experiment, we kept adding URLs to the same list for the entire six day period. For each day, we compute the hit ratio as the number of hits on that day divided by the total number of requests during that day. We also record the size of the SET at the end of each day which is the number of distinct URLs observed by that time. The results are presented in Table 3.1. It can be seen that in all days a high hit ratio of at least 93% is achieved. Also notice that the number of distinct URLs is $\sim 18K$ in the first day and reaches $\sim 54K$ by the end of the sixth day.

Day Number	Hit Ratio	# of Distinct URLs
1	93.4 %	18482
2	95 %	24725
3	97 %	28156
4	94.6 %	38235
5	95.3 %	46565
6	96 %	54040

Table 3.1: Hit ratio and top URL set size that result from running the simple algorithm over University of Calgary's HTTP traces collected between May 1, 2015 and May 6, 2015

Notice that storing the resource lists of such a large number of URLs is feasible because of small size of resource lists (on average 11.7 KB for a page in our top 20 page selection) and abundance of storage space in the remote proxy. However, considering that on average a page can take about 6 seconds to load on a Desktop computer [89], it would take about 33 hours to visit 20000 pages back to back and update their resource lists in metadata repository. Such a long update interval can compromise the freshness of resource lists for some of the fast changing websites.

Notice that the above problem stems from a large number of page requests constantly flowing into the proxy. As a result, we can cast it as an instance of identifying most popular k items in a data stream. Rather than storing all the distinct URLs in a set, in this setting we are interested in an online algorithm that accurately reports top-k elements of a data stream by taking only a single pass over data. This is one of the well-studied problems in data mining community. For instance, authors of [90] presented a survey of some of the most popular algorithms in this area and conducted experiments to compare the performance of these algorithms. Their findings indicate that for insert-only streams[2], the *space saving* algorithm [91] performs better in terms of precision[3], recall[4], used space and update speed. As a result, we select this algorithm for identifying top-k URLs in the stream of URLs arriving at the proxy.

[2]As opposed to streams where elements can be both inserted and deleted
[3]Proportion of the items reported by the algorithm that are true frequent items
[4]Proportion of the true frequent items that are reported by the algorithm

3.2.3.1 Space Saving Algorithm

In order to identify top-k elements of a data stream, the space saving algorithm maintains k elements with their associated counters. Upon arrival of a new URL at the proxy, in case it is already monitored (exists in the list), we just increment its associated counter. Otherwise, if the URL list is not full, we insert the URL into the list and set its counter to 1. If the URL list is full and the URL does not match a monitored item, we find the URL with the least count, *min*, and replace it with the new URL. Finally, *min*+1 is assigned to the count of the new URL. The authors of [91] proposed a data structure called *Stream-Summary* that ensures constant time for finding the minimum element. Also incrementing counters in Stream-Summary can be performed using $O(1)$ pointer operations.

3.2.4 Practical Considerations

Webpage Customization: A growing number of websites provide a mobile version of their content which contains fewer and smaller images and short and concise text [82]. Also browser-dependent code in some webpages can download different set of objects for different browsers [19]. Therefore in order to comply with users' actual needs, the remote proxy needs to be aware of the client attributes such as *user-agent* and device's screen information. To this end, client provides this information to the proxy when it sends the initial request for the page. By using such information, the proxy will be able to imitate the client device when requesting objects from web servers. This way, the proxy can also incorporate the resource list of the corresponding mobile website in its metadata repository.

Incremental Rendering: The bundling feature in WebPro enables the mobile device to stay in low power state during the entire time that the remote proxy fetches the embedded objects of a page. While this can reduce energy consumption of mobile web browsing, it delays receiving the first set of objects by the browser which is required for the partial rendering of the page. To enable drawing intermediate displays in a browser, we can envision

WebPro without bundling in which the proxy forwards each object to the client as soon as it receives the object from a web server. Clearly, such a scheme has the potential to further reduce page load times at the cost of increased energy consumption (compared to WebPro with bundling). We note that implementing WebPro without bundling can benefit from native Virtual Private Network (VPN) support in vast majority of modern mobile devices. Similar to *Meddle* proposed in [92], in this setting, a VPN tunnel can be used to direct all the Internet traffic of mobile device to the remote proxy. Such a VPN-based approach will eliminate the need to deploy a local proxy component on the mobile device.

Cost of Stale Records in Resource List: A webpage's structure can change since the last visit by the profiler which can lead to staleness of some of the records in its corresponding resource list. Considering the superior network connectivity and processing power of the remote proxy, we can ignore the overhead of fetching such stale objects on the proxy. On the other hand, a recent study of object sizes in the top 500 Alexa websites reveals that most of the web objects are typically small to moderate, with the median size being 18 KB [19]. Also because of selective compression component in WebPro, some of those small objects will be compressed before being included in the batch which is usually around a few megabytes for popular webpages. As a result, the overhead of stale objects for mobile device appears as a few extra kilobytes added to the size of a typically large batch file. However, the benefits of WebPro, and specifically elimination of base HTML fetch time, far outweighs such a negligible overhead. On the contrary, a client-only solution may incur significant costs in terms of energy and delay as fetching each of those stale objects can cause state promotion and demotion in the radio of the device, which is a well-known cause of battery drainage on wireless devices.

Profiling Overhead: In WebPro, it is expected that usually the profiler's visit to a page

will occur at an earlier time than serving a user request for that page. However in PBB (the solution proposed in [18,19,59]), each page request triggers a new process of identifying page resources at the proxy. Therefore, in a setting that most webpages already have a corresponding resource list at the proxy, the majority of user requests can be served without incurring any overhead due to profiling.

Handling Asynchronous JavaScript Requests: Most modern webpages use Asynchronous JavaScript requests (AJAX) to dynamically load contents such as advertisements even after the page is loaded (i.e., after the `onload` event). Usually such requests are for session dependent content and hence it would be better to fetch those objects directly from the web servers rather than the proxy. To accomplish this, the local proxy adopts a *selective forwarding* approach in which it forwards the initial page request to the remote proxy and after receiving the page batch from the remote proxy, forwards all subsequent requests to objects not present in its cache to the corresponding web servers.

3.2.5 Prototype Implementation

Our current implementation of WebPro uses the Qt SDK version 5.3. Specifically, QWebKit class which is a result of integration of WebKit into Qt enabled us to develop the web engine component of the system. Also considering that for evaluating WebPro, we compare its performance with PBB, both approaches were implemented using the same Qt libraries. Here we briefly introduce the important parts of our implementation.

3.2.5.1 Resource Profiler

Profiler is responsible for constructing and updating webpage resource lists and storing the metadata information on the remote proxy. The Profiler is basically a WebKit-based web engine which loads webpages on demand. Note that loading a page in the profiler involves all the steps of opening a webpage except rendering. This way, we can obtain the list of all

the objects whether they are resulted from parsing or from JavaScript/CSS evaluations. In particular, we intercept the network activity of this web engine and record the corresponding URLs of all the HTTP requests.

As mentioned in Section 3.2, webpages from the set of popular websites should be loaded periodically in order to keep an up-to-date repository of resource lists on the remote proxy. This is achieved by a bash script that wakes up periodically and iteratively invokes profiler with a URL from a list of top visited websites.

A hash function of the URL determines the unique name and directory of the file that stores its resource list in the repository. In contrast to caching, storage overhead of this approach is negligible because instead of storing actual content of the objects, the proxy stores URLs of those objects. In our experiments, the total space required to store the resource lists of 20 popular websites was about 234 KB. As a result, the entire repository of resource lists can be loaded in the main memory during the operation of the proxy. Disk access is required only for backup purposes.

3.2.5.2 Object Bundling

We use *libtar* library to implement bundling in the remote proxy and unbundling in the client proxy. In our experiments, the time spent in bundling and unbundling is negligible and has a minimal effect on page load times. For example, in the case of an experiment with `www.cnn.com` which contained 139 objects with a total size of 2.6 MB, the time spent in bundling was only 32 milliseconds.

To study the effect of the number of objects on the performance of bundling, we measured the time spent in bundling for different numbers of objects, all with size 20KB (the average object size in a modern webpage). Figure 3.6 depicts the bundling performance as a function of object numbers. It can be observed that even for the case of 200 objects, the bundling time is negligible compared to the overall page load time (in the order of tens of seconds). It is also noteworthy that the timing values reported here are obtained using a typical machine

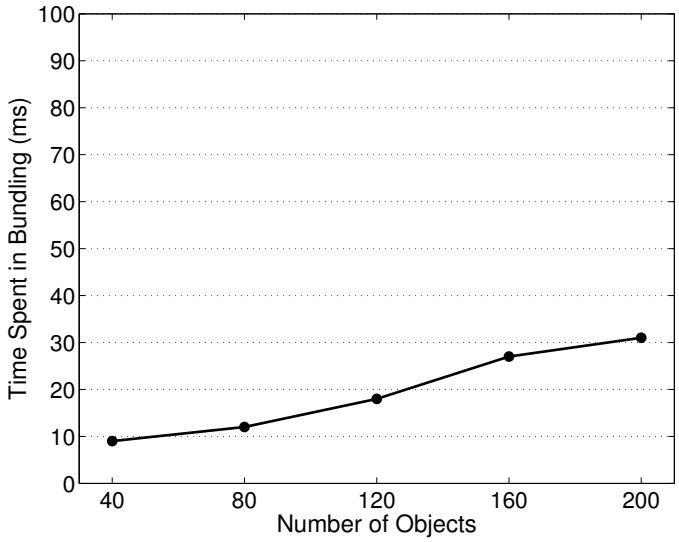


Figure 3.6: Bundling Performance

in our lab, while it is expected that in a real-world deployment, the proxy will be hosted on a more powerful computer(s) with dedicated hardware. With the widespread adoption of cloud computing, we can also envision hosting remote proxy in a cloud platform which automatically scales up its processing power to handle an increase in workload.

3.2.5.3 Selective Compression

According to the results reported in [93], objects that have an image or video `content-type` and also most objects with binary data (e.g. app/octet-stream) already are in compressed form and there is very little room for additional saving. On the other hand, text files such as HTML, XML, JavaScript, and CSS can benefit greatly from compression. In line with this, the remote proxy has a selective compression component that uses the zlib [94] library to compress the body of HTTP responses with the text MIME type. We implemented bundling and selective compression in the same way for PBB as well.

3.2.5.4 Filtering Dynamic URLs

Many websites these days contain references to third party advertisement networks and web tracking systems. Tracking or targeted advertising is done by inclusion of a JavaScript code in a webpage that is executed when a user visits that page. Usually such JavaScript codes use

random numbers or date information to create requests with dynamic URLs (i.e., different URLs over different visits). As a result, the URL generated at the client's browser will be different from the recorded URL at the remote proxy. In other words, these URLs will change at every request and hence the Profiler should avoid recording them. To this end, we have implemented a module in our profiler that filters those changing URLs during the profiling period. In particular, this module detects changing URLs based on the prefixes in URLs and also URLs belonging to a blacklist [95]. To ensure a fair comparison with PBB, we also equipped PBB's web engine in the remote proxy with our filtering module.

Given that the advertisements fetched at different visits to a page can be of varying sizes and/or belong to different domains, we also incorporated the filtering module in our client side proxy to eliminate such variabilities in object load times.

3.2.5.5 Local Proxy

The local proxy is developed using QT's networking API (QTcpSocket and QTcpServer) and acts as a server to the mobile browser while acting as a client to the remote proxy. It also uses the same libraries discussed above for unbundling and decompression. Specifically, local proxy passes the first request of a page to the remote proxy and after receiving the page bundle, responds to the browser with the appropriate object while caching the rest of the objects in the bundle. For all the subsequent requests, local proxy will try to load the object from its cache if available, otherwise will forward the request to the corresponding web server.

3.3 Performance Evaluation

In this section, we use our prototype implementation to demonstrate the effectiveness of WebPro. Notice that we compare WebPro to benchmark system PBB as opposed to conventional web browsers, because the previous work [18, 19] has already shown the superior performance of PBB in comparison to traditional browsers.

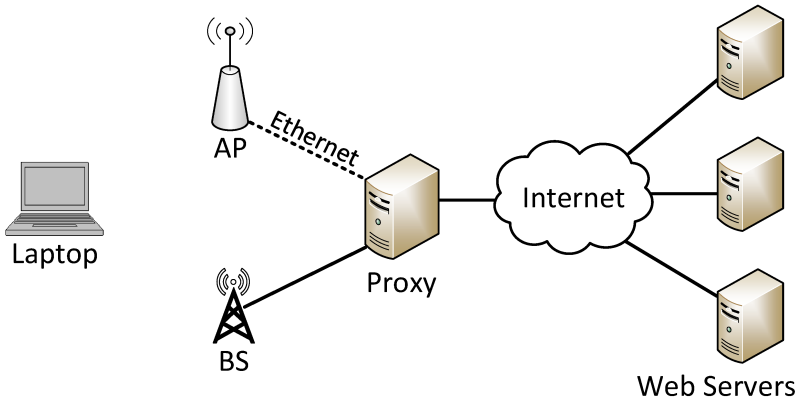


Figure 3.7: Experimental Setup with the Remote Proxy.

3.3.1 Experimental Setup

Client Setup: Figure 3.7 depicts our experimental setup. We chose an ASUS UX31A laptop running Ubuntu 14.04 with built-in WiFi adapter as our mobile terminal. For cellular measurements, we equipped the laptop device with an LTE USB modem so that it can access the LTE network provided by a major Canadian cellular carrier. As mentioned in [1], the rationale for using laptops instead of smartphones is that slower processors of smartphones can influence our results on page load times. Also, by using laptops, we don't have to restrict our experiments to those websites that provide a mobile version of their site.

On the client side, we developed our own browser using QWebKit library. This way we can log detailed timing information and also clear browser's cache programmatically before each experiment. In practice, any browser can benefit from our proxy-based solution without any modifications. It only requires configuring the browser to use the local proxy.

Infrastructure Setup: We performed WLAN measurements using a *Cisco Linksys* EA2700 wireless router. The router was connected to the proxy server through the campus LAN (100 Mbps Ethernet). We also conducted cellular experiments over the LTE network at a location with good signal strength. In the cellular setting, the proxy was configured with a public IP address. The average TCP throughput between the mobile device and the remote proxy, measured by `iperf` tool, was about 52.5 Mbps and 2.5 Mbps in WiFi and Cellular

Webpage	Size (KB)	# of images	# of JS	# of CSS	# of other	Total # of objects
cnn.com	2712	90	36	1	12	139
espn.go.com	2404	76	13	3	5	97
mozilla.org	957	18	5	3	7	33
walmart.ca	3239	51	12	3	4	70
bbc.com	1599	43	24	3	3	73
ebay.ca	4078	132	4	3	9	148
shaw.ca/store	1944	26	20	2	10	58
go.com	3224	22	30	8	16	76
nytimes.com	2974	84	40	8	9	141
deviantart.com	2102	68	14	4	2	88
apple.com	1254	25	18	6	1	50
ikea.com/ca/en	2923	56	13	5	3	77
flickr.com	6736	24	4	2	8	38
ca.ign.com	2473	62	24	13	6	105
microsoft.com	1208	35	10	1	7	53
homedepot.ca	2180	32	12	5	11	60
Wikipedia Article	1932	80	9	2	1	92
cbssports.com	1535	37	26	2	5	70
tripadvisor.ca	3510	78	5	1	4	88
about.com	1437	43	5	2	3	53

Table 3.2: Characteristics of the Websites Used in the Experiments (accessed on Oct. 29, 2014).

settings, respectively. Also the average ping RTT between the mobile device and the remote proxy was about 10 ms and 117 ms in WiFi and Cellular settings, respectively. The remote proxy was hosted on a fairly typical machine running Ubuntu 14.04 with no special server capability. This machine is connected to Internet using a 100 Mbps LAN connection. All experiments were conducted in a lab environment.

3.3.2 Workload Characterization

We selected 20 webpages from the top Canadian websites listed on Alexa [84]. Similar to [1], we used desktop versions of these websites instead of their mobile versions because of widespread use of tablets and large screen smartphones. These webpages were chosen from different categories such as news, auction, sports, shopping, etc. Table 3.2 shows the detailed properties of our selected webpages. The average page size is 2521.05 KB and the total number of objects ranges from 33 to 148. Anything other than image, JavaScript and CSS is counted as *other*.

3.3.3 Performance Metrics Used

Page Load Time: We use page load time (PLT) as the primary indicator of user-perceived performance. In our measurements, page load time is the time elapsed between the initial page request and the time when all associated objects of a page have been downloaded and processed. This time is identified by the occurrence of the `onload` event at the browser and includes the time spent in executing CSS and synchronous JavaScript files. In the proxy-based systems discussed here, PLT consists of the following components:

1. Time to request the page from the remote proxy,
2. Time to download all the objects in the resource list (in WebPro) or the time it takes for the remote proxy's web engine to load the page (in PBB),
3. Time to receive the bundle from the remote proxy, and,
4. Time to download all the objects that are missing in the bundle until the entire webpage is loaded.

Hit Ratio: In order to capture the amount of change in webpage structures, we use the *hit ratio* metric. The hit ratio associated with a webpage's resource list is the number of objects from the resource list that are actually requested during the page load process, divided by the total number of objects in that resource list. It represents the fraction of the resource list that is still valid and accurate. A high hit ratio means that there has been little change in webpage's structure since the last time that the profiler visited the page.

3.3.4 Measurement Results

3.3.4.1 Change in Webpage Structures

The underlying hypothesis in WebPro is that the resource structure of a website changes less frequently than the actual content of the objects and webpages. We note that web publishers usually choose a short expiration time for web objects and also prevent web resources from being cached by using "`no-store`" in the `cache-control` HTTP header field.

In line with this, our first experiment studies the temporal changes in webpage structures. In particular, it monitors the average hit ratio of the resource lists of the websites presented in Section 3.3.2. As mentioned in Section 3.3.3, a decline in the value of the hit ratio associated with a resource list corresponds to change in that page's structure. Note that our selected webpages are a combination of fast changing pages such as news websites as well as stable homepages of large companies such as Apple.

We conducted five experiments over the span of five weeks, each separated by one week. In each experiment, we first constructed the resource list of the webpages and then used them to load the same pages every hour over an 8 hour period. Figure 3.8 plots the average hit ratio and 95% confidence intervals of the webpages among all the experiments as a function of the hours passed since loading the page for the first time. We see that the highest amount of hit ratio is achieved in the first hour, as expected. It can be observed that the amount of change in webpage structures over an eight hour period is relatively low. The difference between the average hit ratio in the first and eighth hours is less than 0.1 and the maximum amount of hour to hour change in the average hit ratio is about 0.02. As a result, it should be feasible for the remote proxy to capture the temporal changes in webpage structures by updating its resource list repository in a timely manner (every three hours in our experiments). Notice that our selected three hour update interval is even less than the 4 hour update interval proposed by [10] for capturing the flux in dependency structure of webpages.

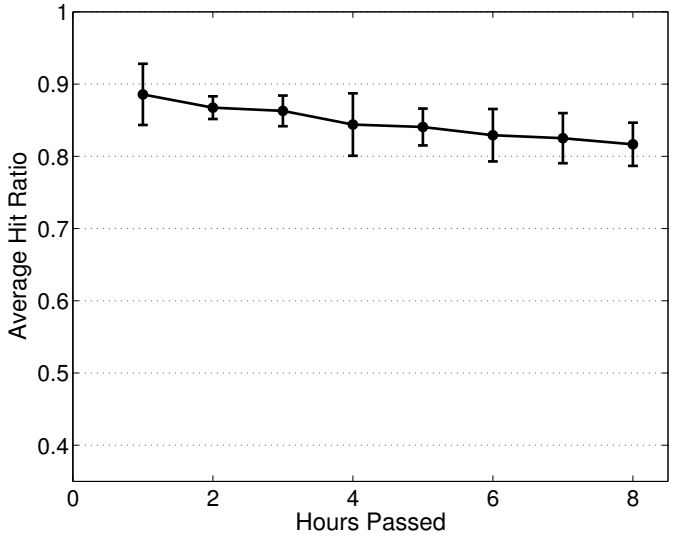
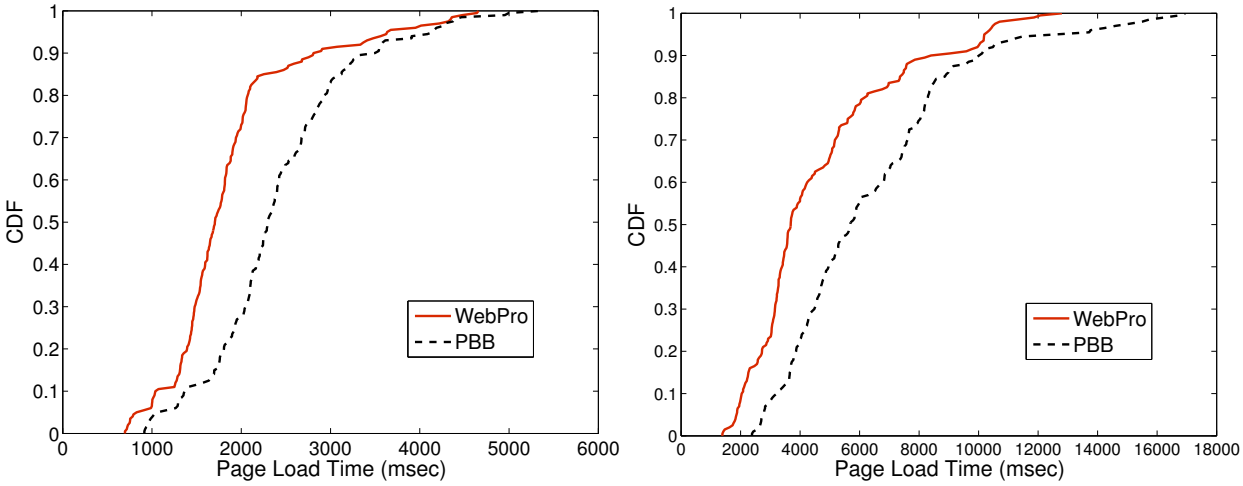


Figure 3.8: Temporal Change in Webpage Structures. Drop in the value of the average hit ratio over time is an indication of the change in the structure of the webpages. However, the amount of such change is relatively low over an eight hour period.

3.3.4.2 Comparison with Benchmark

Next, we compare the performance of WebPro and the benchmark PBB, using the webpages presented in Section 3.3.2. Because of the variability in load times between consecutive page visits, we performed ten back to back experiments with each page. Our experiments were conducted during quiet times and the browser's cache was cleared programmatically before each experiment. Speculative loading at the proxy involves using resource lists associated with user-requested webpages and in our experiments, the remote proxy used the resource lists that were constructed three hours before the actual measurements. Given the abundance of computation and communication resources at the proxy, it is feasible for the proxy to update its resource list repository of top visited webpages every three hours. Moreover, the results of our experiment in the previous section show that the amount of change in webpage structures within three hours is negligible.

Figure 3.9 represents the cumulative distribution function of page load time under these two approaches in WLAN and cellular settings. It can be seen that WebPro performs better in terms of page load time. Figure 3.9(a) shows that in the WLAN environment, WebPro



(a) WLAN Measurements

(b) Cellular Measurements

Figure 3.9: Cumulative Distribution Function of Page Load Time. WebPro outperforms benchmark PBB. In the WLAN setting, under WebPro, 73% of the pages load in less than 2 seconds. However, in the PBB approach, 28% of the instances complete loading within 2 seconds. In the cellular environment, under WebPro, 78% of the page loads complete within 6 seconds while under PBB, only 55% of the pages complete loading in that time.

Webpage	Page Load Time (ms)		Improvement
	PBB	WebPro	
www.tripadvisor.ca	3835	3623	5.5%
www.deviantart.com	10675	10070	5.7%
www.flickr.com	6717	3278	51.2%
www.about.com	4456	2166	51.4%

Table 3.3: Improvement in Average Page Load Time.

helps up to 73% of the pages to load in less than 2 seconds, while with PBB only 28% of the instances complete loading in that time. Similarly Figure 3.9(b) shows that in the cellular environment, under WebPro, 78% of the pages finish loading within 6 seconds, but under PBB, only 55% of the instances finish loading in that time. In general, across all the experiments performed in the WLAN and cellular environments, our results indicate that an average of 26% reduction in page load times can be achieved by using WebPro. Figure 3.9(b) also confirms that in cellular networks, the same webpages experience longer load times underscoring the importance of page load time reduction in such networks.

Table 3.3 zooms into the details of these measurements by listing two of the webpages with the lowest amount of improvement and two of the pages with the highest reduction

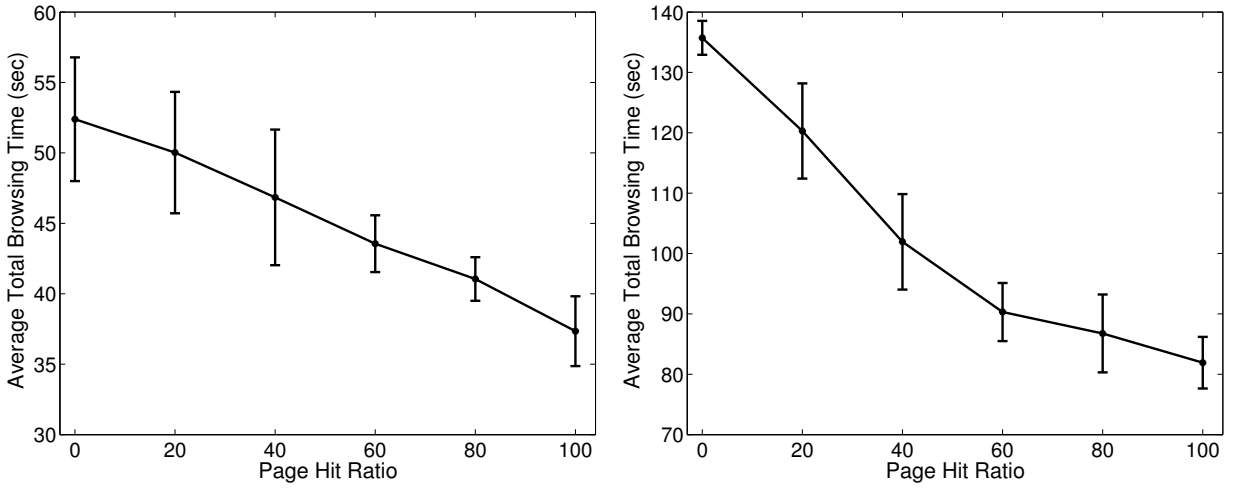
in load time. It shows that the improvements can range from 5% to 51%. Note that the variability in improvement across websites results from several factors, of which we mention only a few:

- The number of domains that web objects are spread across which affects the number of unique connections required to fetch all the objects.
- The size of the website as indicated by the total number of bytes and also the number of objects.
- Website design which creates different set of dependencies between operations of the page load process [26]. This can impose different orders for retrieving web objects.
- Topological proximity between the client and original web server or an edge server from content distribution networks (CDNs).

3.3.4.3 Effect of Page Hit Ratio

In a real deployment, it is possible that the remote proxy will not have the resource lists associated with all the user requests. In that case, it will load the page in a web engine and will send the whole page in a bundle to the client. That is, the remote proxy will employ a combination of the web engine-based and speculative loading approaches to satisfy user requests.

In light of this, our next experiment evaluates the improvements in page load time in a more realistic scenario. Here we gradually increase the hit ratio for the test webpages, that is we increase the fraction of user requests with a corresponding resource list at the proxy. To distinguish this fraction from the hit ratio metric introduced in Section 3.3.3, we call it *page hit ratio*. Using the same webpages presented in Section 3.3.2, we conducted five experiment runs associated with each page hit ratio. At each run, the remote proxy uses resource lists for a random set of pages that are determined based on the page hit ratio, and employs ordinary page loading for the rest of the pages. As a clarifying example, assume



(a) WLAN Measurements

(b) Cellular Measurements

Figure 3.10: Back to Back load time for 20 popular webpages as a function of page hit ratio. An increase in the page hit ratio reduces the total browsing time. In the case of WLAN and cellular measurements, there is a maximum reduction of 28% and 39%, respectively. The maximum improvements are achieved at 100% page hit ratio.

that the remote proxy is going to serve 20 distinct page requests. In the case of 40% page hit ratio, for each run, proxy randomly selects 8 out of the 20 pages to load using resource lists and employs web engine for loading the remaining 12 pages.

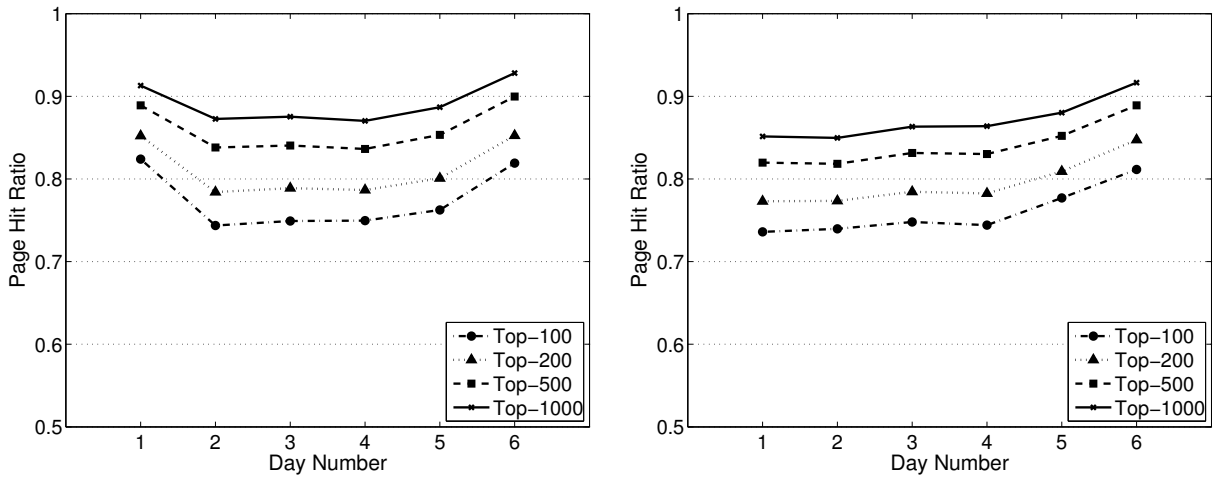
Figure 3.10 shows the average value for the total time to visit all 20 webpages back to back as a function of the page hit ratio. The results are represented with 95 percent confidence intervals. It can be seen that a higher page hit ratio leads to a greater improvement in user's browsing experience. The upper bound of reduction in back to back page load time is 28% and 39% in the case of WLAN and cellular measurements, respectively. These upper bounds correspond to a 100% page hit ratio in both experiments.

From Figure 3.10 we can see that the amount of improvement gained from using resource lists depends on the page hit ratio. We observed in section 3.2.3 that adding all the new URLs to profiler's URL list can result in high page hit ratios that are close to 100%. However, this may lead to a long update interval in profiler and endanger freshness of resource lists. The immediate alternative is to keep a relatively small summary of data stream (stream of URLs) rather than storing all of them. As discussed before, we propose using the space

saving algorithm to identify top-k popular URLs of the stream.

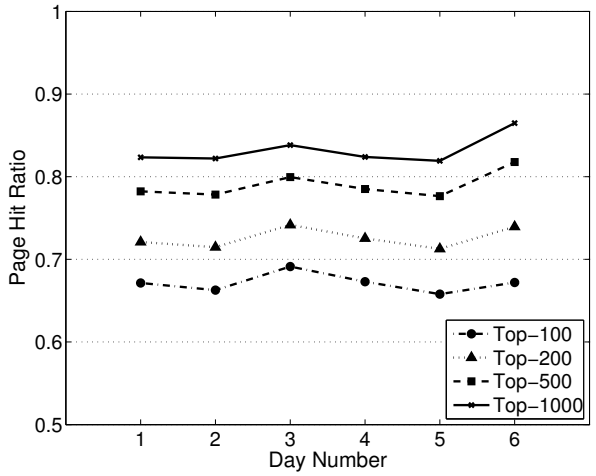
To study the effectiveness of this approach in a real setting, we ran the space saving algorithm over network traffic traces collected from the University of Calgary's Internet link. Similar to the experiment explained in section 3.2.3, we used an `AWK` script to extract URLs of the landing pages from HTTP traces and fed them to the space saving algorithm. We performed four experiments over four different six-day intervals. At each experiment, we constructed the top-k list for the URL stream of six consecutive days while measuring page hit ratio separately for each day. Specifically, with each URL in the stream, we first check the top-k list to determine whether it is among the current popular elements (hit occurrence), and if so, we increment both its associated counter in the list and a hit counter. Otherwise, we add the URL and its associated counter to the top-k list in accordance with the space saving algorithm. Finally, page hit ratio of each day is calculated as the number of hits during that day divided by the total number of page requests received on that day.

Figure 3.11 shows the page hit ratios achieved for different values of k. It can be seen that by increasing the size of the popular URLs list, k, the page hit ratio increases. For example, in May 25-30 period, by increasing k from 100 to 1000, the average page hit ratio increases from 67% to 83%. Figure 3.11 also shows that in all four time periods, by keeping a popular URL list with only 1000 items, a page hit ratio of above 80% can be achieved. By assuming an average 6 seconds page load time on a Desktop computer [89], updating the resource lists of 1000 websites at the proxy will take only one hour and forty minutes. Comparing such a short update time to our results in section 3.3.4.1 on the frequency of change in webpage structures implies that the profiler would be able to capture temporal changes in the structure of top-1000 popular webpages (a subset of all the requests in the network) and still provide a high page hit ratio to its users.

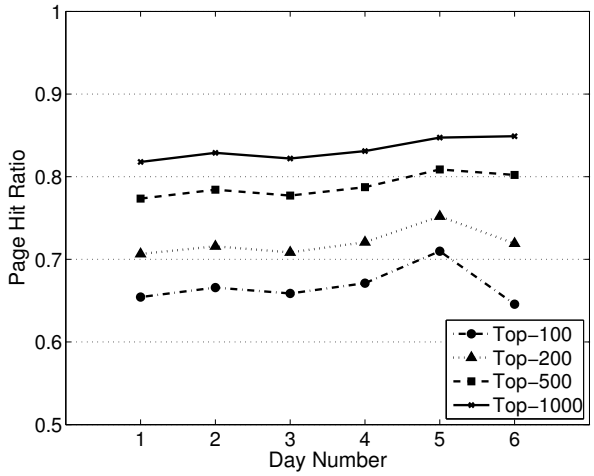


(a) January 5-10, 2015

(b) February 09-14, 2015



(c) May 25-30, 2015



(d) June 8-13, 2015

Figure 3.11: Page Hit Ratios achieved by applying the space saving algorithm to the HTTP traces collected from the University of Calgary's Internet link over four different six day intervals. Increasing the size of the popular URLs list leads to higher page hit ratios. Also, on average, keeping just the top-1000 popular URLs in the stream of URLs that arrive at the proxy, results in over 80% page hit ratios.

3.3.4.4 Effect of Concurrent Connections

As mentioned in Section 3.2, WebPro uses multiple concurrent connections to fetch all objects in the resource list associated with a webpage. Similarly, typical web engines use concurrent TCP connections to avoid the head-of-line blocking problem and reduce page load time [42]. However, in modern web engines there is a limit on the number of concurrent connections per domain. For example, the Chrome browser on Android mobile operating system limits the number of simultaneous connections per domain to 6. The WebKit-based web engine used in our implementation also caps the number of parallel connections per host/port combination to 6. This limitation is imposed by Qt's network access manager class and hence it is also applied to our implementation of WebPro, which uses the same class for network operations.

Our next experiment studies the effect of the number of concurrent connections on the performance of WebPro and PBB. Figure 3.12 shows the average page load time for a Wikipedia article page under a varying number of maximum concurrent connections. The results are averaged over 10 runs and error bars represent 95% confidence intervals. We observe a significant performance improvement in both approaches by increasing the number of concurrent connections. Specifically, increasing the concurrency limit from 2 to 8 results in 48% and 23% faster page load time in the case of WebPro and PBB, respectively. The justification for better performance of WebPro is that an increased number of concurrent connections allows more subresources to be fetched in parallel.

We also found that increasing the concurrency limit beyond 6 leads to marginal improvements in page load times. This can be due to several factors creating a bottleneck for browsing performance. For example, by increasing the concurrency beyond a limit, each connection obtains less bandwidth, which results in longer delays when downloading objects. On the other hand, high concurrency requires more TCP connection states and buffers to be maintained at the remote proxy and hence increases the processing overhead on the proxy.

Figure 3.12 also shows that WebPro benefits more from increased concurrency, compared

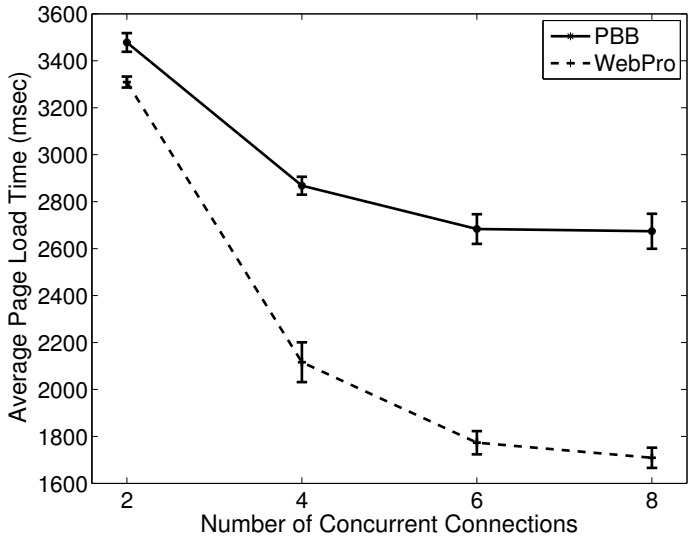


Figure 3.12: Average Page Load Time for a Wikipedia article page as a function of the number of parallel connections. We see that increasing the concurrency reduces the page load time. The benefits are greater for WebPro as it can fetch more subresources concurrently.

to the PBB approach. In particular, a 4% difference in page load time between two approaches reaches 36%, by increasing the concurrency restriction from 2 to 8. This is due to the fact that processing tasks such as JavaScript evaluation can serialize the page load process in PBB's web engine. However, by using the resource list of a webpage, WebPro can utilize the full potential of concurrent connections.

3.3.4.5 Effect of Network Delay

As mentioned in Section 3.2, WebPro improves the performance of mobile web browsing by eliminating the initial RTT required to fetch the base HTML file of a webpage. The length of this time varies depending on the distance between the remote proxy and web servers, and the type of networks involved. Other factors such as queuing delays or congested links can also contribute to the variability in the end-to-end delay between the proxy and web servers. In order to study the impact of network delay on page load time, we conducted a set of experiments by artificially controlling the amount of packet delay in our tests.

We used the *dummynet* network emulator [96] to inject extra delay between the remote proxy and web servers. Specifically, we added 100, 200, 300 and 400 ms extra delay to

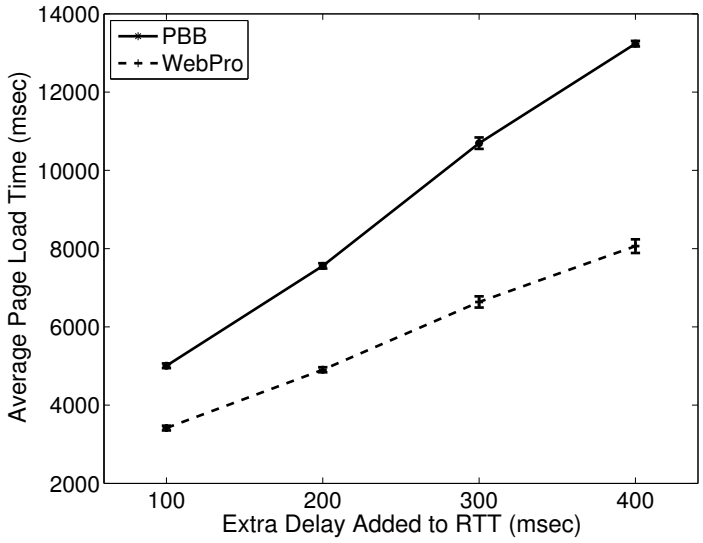


Figure 3.13: Average Page Load Time for a Wikipedia article page as a function of network delay. We see that higher RTT values lead to higher page load times. By increasing RTT, PBB incurs higher latencies compared to WebPro.

the round trip time between our device and the servers hosting the objects referenced in a Wikipedia article page. Figure 3.13 shows the average page load time under WebPro and PBB as a function of the network delay. The results represent the average of ten runs along with the 95% confidence intervals. It is observed that increasing RTT (i.e., network delay) leads to a slower browsing experience in both approaches. In particular, raising the amount of injected delay from 100 ms to 400 ms increases the average page load time by 136% and 164% in WebPro and PBB, respectively.

Figure 3.13 also shows that with larger RTTs, the amount of savings achieved by WebPro increases. This can be explained by the notions of *dependency graph* and *critical path*, introduced in [26]. The dependency graph of a webpage is a directed acyclic graph with load process activities as nodes. The edges of this graph represent the dependencies between those activities. Given that each node is associated with the duration of completing its corresponding activity, the simplest form of critical path is defined as the longest path in the dependency graph. Since in PBB, the extra delay impacts all the resource loading nodes of a critical path, the overall page load time will be affected by the aggregate of those extra

delays. However, WebPro avoids traversing the critical path by downloading the objects in the resource list of a page.

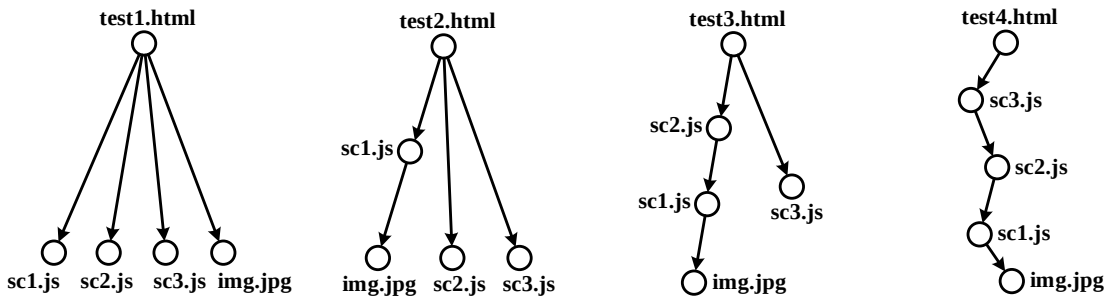
3.3.4.6 Effect of Webpage Complexity

As mentioned in section 3.2.2, there are inter-object dependencies in today's webpages that lead to the serialization of network transfers required for loading a page. One of the common cases of such dependencies is created when an embedded object itself embeds other objects. For example a JavaScript object can embed any kind of object and a CSS file can embed background images. In this case, discovering the object referenced in a script file, requires another RTT between the browser and the origin server. Because of such dependencies, a browser (or a web engine) cannot discover all the embedded objects of a page right after fetching and parsing the base HTML file. On the contrary, resource exploration becomes an iterative process in which local computations such as parsing and script executions are interleaved with network transfers [23].

Other than eliminating the base HTML fetch time, one other reason for WebPro's superior performance is that it breaks such object level dependencies by using a previously recorded resource list. To study the benefits that come from eliminating inter-object dependencies, we carefully designed 4 webpages, all with the same set of embedded objects. The base HTML files of these pages have slight differences but all are of the same sizes (174 Bytes)[5]. Also in all the 4 pages, the final rendered page is the same which consists of just a pigeon image on the screen. The major difference between these pages is in the amount of dependency between their objects. Specifically, from the first page (test1.html) to the last page (test4.html), we gradually increase the length of the critical path in their dependency graphs. Figure 3.14 depicts the dependency graphs for these 4 pages. Test webpages are available at http://pages.cpsc.ucalgary.ca/~asehati/webpro/.

We hosted our test pages on an Apache web server running on a Linux machine in our lab.

[5]The size of the HTML files were made equal by inserting the required number of blank spaces after the </html> tag.



(a) level 1 (b) level 2 (c) level 3 (d) level 4

Figure 3.14: Dependency graphs for four carefully designed test pages with the same set of embedded objects. In the first test page (a), all the objects can be discovered after fetching and parsing the base HTML file, giving it a critical path of length 1. The second page (b) has a critical path of length 2, because the image object can be revealed after fetching and evaluating the JavaScript object sc1.js. In the third page (c) with critical path length 3, fetching and evaluating JavaScript object sc2.js, reveals another JavaScript object, sc1.js, the fetching and evaluation of which reveals the image object. Finally, in the last page (d) with critical path length 4, evaluating sc3.js reveals sc2.js, evaluating sc2.js reveals sc1.js and evaluating sc1.js reveals the image object.

Similar to the previous section, dummynet was used to inject 200 ms emulated delay between remote proxy and the web server. Using the same WiFi setting described in section 3.3.1, we loaded each test page ten times with WebPro and PBB and computed the average speedup achieved with WebPro[6]. Borrowing the definition from [97], WebPro's speedup relative to PBB, is the ratio of page load time using PBB to the page load time under WebPro. Given that all the 4 pages have the same set of embedded objects and their base HTML files are of the same size, WebPro results in the same page load times for all of the test pages. The reason is that the resource list files of all the pages point to the same set of embedded objects hosted on the same server and also base HTML fetch times are the same. However, different levels of complexity in these pages lead to different page load times under PBB which by using a web engine goes through an iterative process of discovering embedded objects.

Figure 3.15 depicts WebPro's speedup relative to PBB as a function of critical path length under two settings. In one setting (called *first setting*), persistent connections were supported by the web server and in the other setting (called *second setting*), persistent connections were disabled in the web server. Notice that for a given test page, WebPro achieves the same load

[6]95% confidence intervals were also computed but are not presented since they were very small.

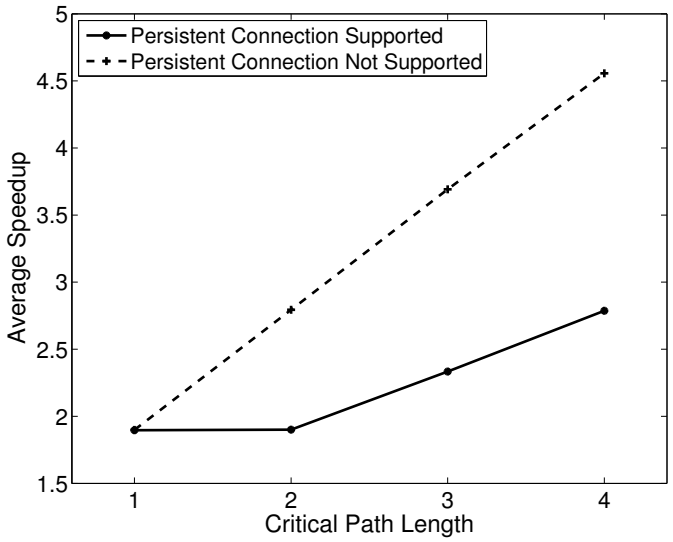


Figure 3.15: Speedup of WebPro relative to PBB as a function of critical path length. As the critical path becomes longer, the speedup with WebPro increases. Also, for a given webpage, speedups with WebPro are higher under the setting that does not support persistent connections.

times under these two settings, but in three out of the four test pages (test 2 through 4), PBB achieves higher page load times under the second setting. The reason is that WebPro's proxy establishes 5 parallel TCP connections to the server right at the beginning and by using each connection only once, does not get affected by the lack of support for persistent connections. However in PBB under the second setting, every time that evaluating one object reveals another object, the proxy will incur one extra RTT to establish a new TCP connection for fetching the newly discovered object. For these reasons, we can see in Figure 3.15 that for a given webpage, speedups with WebPro are higher if persistent connections are not used between the proxy and the web server. The only exception is page 1 in which all the objects are referenced in the base HTML and there are no inter-object dependencies.

From Figure 3.15, it can also be observed that speedups with WebPro increase as the critical path becomes longer. By increasing the critical path length from one to four, the speedup of 1.9 reaches 2.79 and 4.56 in the first and second settings, respectively. Given that WebPro achieves the same page load times for all pages, higher speedup comes from higher page load times under PBB. Specifically, in the first setting, increasing the critical

path length by one adds one RTT to the page load time of PBB which is the time required to fetch and evaluate the new referencing object inserted in the critical path. In the second setting, extending the critical path by one edge adds two RTTs to the page load time of PBB. One RTT is incurred for establishing a new TCP connection to the server and another RTT is incurred for fetching and evaluating the new referencing object.

We observe that in Figure 3.15, under the first setting, WebPro achieves the same speedups for page 1 and 2 which again is due to the fact that same page load times are achieved with PBB for these two pages. Figure 3.16 shows the waterfall of loading these two pages using PBB under the first setting. To load test1.html, Web engine first establishes a connection to the server (we call it connection 1) to fetch the base HTML file of the page. After fetching and parsing the base HTML file, the web engine finds links to four new objects (img.jpg, sc1.js, sc2.js, and sc3.js). Because of persistent connections, connection 1 is still available and can be used for fetching one of the four newly discovered objects. Specifically, web engine issues the request for sc1.js over connection 1 and at the same time initiates three new parallel connections to the server (we call them connections 2, 3 and 4). Notice that initiating a connection means the exchange of SYN and SYNACK segments between the proxy and the server which takes one RTT. On the other hand, considering the small size of sc1.js (65 bytes), it takes one RTT to request and receive this file at the proxy. As a result, by the time that those three connections are established, sc1.js has arrived at the proxy and connection 1 has become available again. Therefore, at this point in time (marked as 600 ms in Figure 3.16(a)), the proxy has 4 available connections to the server (connections 1, 2, 3 and 4) but there are just 3 objects remaining from the page (img.jpg, sc2.js and sc3.js). The proxy proceeds by using connection 1 for fetching img.jpg and at the same time issues requests for sc2.js and sc3.js over two of the three newly opened connections (connections 3 and 4). Finally, the proxy is able to fetch all the required objects of the page without using connection 2. A similar explanation can be used to describe waterfall of page test2.html

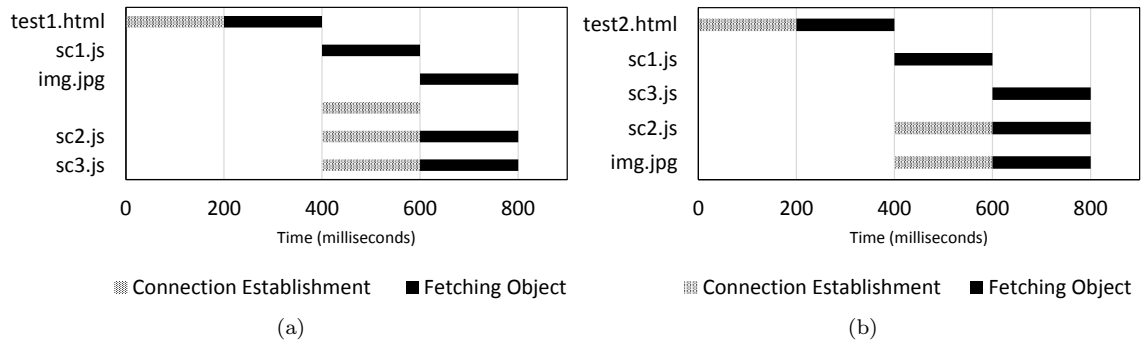


Figure 3.16: Waterfall of loading (a) the first and (b) the second test page using PBB with persistent connections enabled. In (a), test1.js, sc1.js and img.jpg and in (b) test2.html, sc1.js and sc3.js are downloaded over the same connection.

which is downloaded at the proxy after four RTTs.

We note that the last three experiments study the behaviour of WebPro and PBB under different system conditions, i.e. concurrency limit, network delay and webpage complexity. Given that these conditions only affect the wired part of the network between the remote proxy and web servers, we only presented the experimental results under WLAN setting. Similar behaviour is expected in the cellular environment.

3.3.4.7 Energy Impact of Bundling

As mentioned in section 3.2, the remote proxy sends all the required objects of a page in a bundle to the client. Transmitting a batch instead of a sequence of small objects prevents the radio of the device from constant promotions and demotions which can quickly drain the battery of the device. On the other hand, as mentioned in section 2.4 (Incremental Rendering), bundling is not an indispensable feature of WebPro and we can envision WebPro without bundling.

However, to verify the energy efficiency of WebPro with bundling, we performed a set of experiments using real webpages. To do so, we used most of the experimental setup described in sections 3.3.1 and 3.3.2 of this chapter. The only difference was that we emulated LTE network conditions using dummynet network emulator. Specifically, dummynet was used to inject extra delay and throttle bandwidth so that the RTT and throughput of our emulation

Webpage	Radio-on Time (ms) WebPro w/ bundling	WebPro w/o bundling	Improvement
business.gov.au	2333	3395	31.3%
spark.co.nz	3229	3881	16.8%
mashreghnews.ir	3042	5225	41.8%
zju.edu.cn	3374	4521	25.4%

Table 3.4: Improvement in Average Radio-on Time with Bundling.

would be consistent with real-world settings reported in recent measurement studies [98]. In our experiments we measure the radio-on time which starts from the time that the client receives the first set of bytes and ends with the reception of the last byte. In order to consider LTE's Radio Resource Control (RRC) state machine, we also incorporated tail time effect in our analysis. Specifically, idle gaps between consecutive objects that are greater than the tail time, only contribute the amount of tail time to the radio-on time.

Table 3.4 presents our results for 4 different webpages that are among popular sites in 4 different countries. All measurement results are averaged over 10 runs. It can be seen that bundling reduces radio-on time which implies reduction in energy consumption of mobile web browsing. However, the amount of improvement achieved with bundling varies between different webpages. This is due to a set of reasons such as different amounts of inter-object idle gaps, different number of objects and differences in topological distance between the proxy and the web servers.

3.4 Discussion

User Mobility: While we addressed the challenges arising from long access delays of wireless networks, it should be noted that the mobility of users while surfing the web can make accelerating mobile web even more challenging. Specifically, mobility of users can cause unpredictable network conditions, rate variability and signaling overheads, all of which can contribute to the poor browsing experience of mobile users. Considering the recent efforts on the application of Multi-Path TCP (MPTCP) to mobile devices [99], mobility of users

can be facilitated by using multiple interfaces (WiFi and cellular) available in most of the mobile devices. In such a setting, MPTCP's backup mode [100] will be used to provide the user with a seamless transition experience as he/she walks between WiFi APs or walks out of the WiFi coverage.

WebPro in Error-Prone Wireless Networks: In WebPro, the transmission of bundles between the client and the remote proxy is performed over TCP. This implies that the TCP protocol will provide a reliable channel service to our system, WebPro. Specifically, TCP will react to any packet loss by retransmitting the missing packets. In other words, the granularity of TCP's retransmission is in the packet level and from this perspective, there is no difference between transmitting individual objects versus transmitting bundles over the wireless link.

Chapter 4

Energy-Delay Tradeoff for Request Bundling on Smartphones

To reduce the energy consumption of mobile web browsing, in Chapter 3, we employed the bundling technique. Specifically, in WebPro, the mobile device receives all the required objects of a webpage in one bundle from the remote proxy. This approach can reduce the time the radio interface is on, by eliminating the idle gaps between web objects. However, using a single bundle during the load process, trades off increased page load time (compared to WebPro without bundling) for the potential to achieve maximum possible energy saving. This is due to the fact that in this scheme, the remote proxy should wait to receive all the embedded objects of a page before it can create the bundle and send it to the client. This implies that the bundling scheme used in Chapter 3 is capable of achieving only a fixed point on the energy delay tradeoff associated with bundling. It is also worth noting that mobile web browsing is not the only application that can benefit from bundling. As stated in Chapter 1, bundling can reduce radio energy consumption in other contexts such as mobile code offloading and delay tolerant applications.

Clearly, the side effect of bundling is the additional delay experienced by application users. Given that different users may have different preferences for energy versus delay, it is desirable to devise systematic solutions that have the flexibility in achieving different energy delay tradeoffs. Thus, in this chapter we study a formal notion of optimal request bundling in the general context of data traffic in cellular networks. Specifically, we formulate bundling as a cost minimization problem, in which the tradeoff between energy and delay is captured by a cost function. We then propose an online algorithm that can solve the problem without knowing the future data transfer requests a priori. As a benchmark for the evaluation of

our algorithm, we design a dynamic programming-based optimal offline algorithm that relies on the *unrealistic* assumption of knowing the entire sequence of the data transfer requests in advance. We perform competitive analysis of the online algorithm by comparing it to the optimal offline algorithm. This is followed by performance evaluation of the proposed algorithm in a range of realistic scenarios using both model-driven simulations and real experiments on a smartphone.

The rest of the chapter is organized as follows. We start by formally describing the problem in Section 4.1. In Section 4.2 we present an optimal offline algorithm. Our proposed online algorithm is presented in Section 4.3, which is then followed by a detailed analysis of the algorithm in Sections 4.4 and 4.5. Performance evaluation results are discussed in Section 4.6.

4.1 Problem Statement

Consider a sequence of data transfer request arrivals $\mathcal{A} = \langle a_1, \ldots, a_n \rangle$, where a_i denotes the arrival time of request i. *The sequence $\mathcal{A}$ is not known in advance.* Without loss of generality, we assume that the radio is off when the first request arrives. The goal is to design an online algorithm to bundle multiple requests together and grant them at once as opposed to individually granting each request. Depending on the application context, a data transfer request may involve uploading and/or downloading data over the radio interface.

Let $\mathcal{G}_A = \langle g_1, \ldots, g_k \rangle$ denote the sequence of grants made by some algorithm A, for the arrival sequence $\mathcal{A}$, where g_i denotes the time of grant i. Let $\mathcal{X}_A = \{X_1, \ldots, X_k\}$ denote the set of all grant intervals of algorithm A, where $X_1 = [a_1, g_1]$ and $X_i = (g_{i-1}, g_i]$, for $i \geq 2$. All requests that arrive during the interval X_i are bundled together and granted at time g_i. Throughout the chapter, we use the notation X_i to refer to the i-th grant interval as well as the length of that interval, when there is no ambiguity.

Fig. 4.1 shows the relation between arrivals and grants. The objective of the bundling

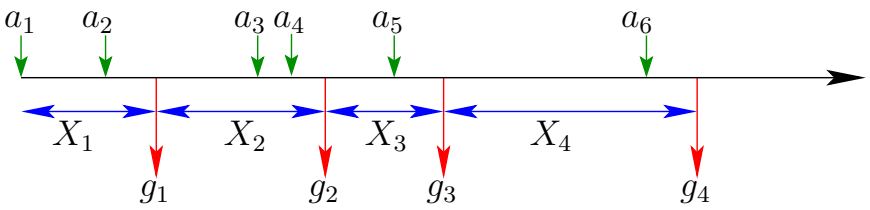


Figure 4.1: Relation between arrivals, grants and intervals.

algorithm is to determine the grant times g_i that minimize the cost $C_A = E_A + \alpha D_A$, where, C_A denotes the total cost of algorithm A. E_A and D_A denote the *energy cost* and *delay cost* of algorithm A, respectively. The weight factor $\alpha \geq 0$ is used to achieve a desired tradeoff between energy and delay. A smaller value of α indicates the willingness of the user to tolerate a higher delay for the sake of reducing the radio energy consumption. By controlling α, different energy-delay tradeoffs can be achieved ranging from maximum energy reduction (with $\alpha = 0$) to zero energy saving (with $\alpha \gg 1$).

4.1.1 Energy Cost

The energy cost E_A is the tail energy consumed because of inactivity periods between grants of algorithm A. Let T denote the tail time. We use the following function to characterize the tail energy:

$$\varepsilon(\tau) = \min\{\tau, T\} \tag{4.1}$$

where, τ is the time passed since the last grant of the algorithm (see Fig. 4.2). Then, the energy cost of grant interval X_i is given by $E_A(X_i) = \varepsilon(X_i)$. It then follows that,

$$E_A = \sum_{X_i \in \mathcal{X}_A} E_A(X_i) + T = \sum_{X_i \in \mathcal{X}_A} \varepsilon(X_i) + T, \tag{4.2}$$

where the additional term T is added to account for a tail time after the last grant. In other words, we define E_A as the time the radio spends in the on state under algorithm A. To simplify the analysis, similar to [16, 74, 76], we have ignored the transfer time of bundles as this time is the same for every bundling algorithm.

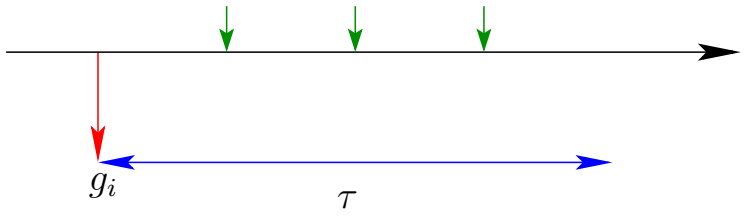


Figure 4.2: τ is the time since the last grant.

4.1.2 Delay Cost

The delay cost of the algorithm is defined as the sum of delay costs of all the bundles. We use the notation $D_A(X_i)$ to denote the delay cost of bundle i, which includes all requests that arrive during interval X_i. Then, we have,

$$D_A = \sum_{X_i \in \mathcal{X}_A} D_A(X_i)\,.$$

Consider a request $a_j \in X_i = (g_{i-1}, g_i]$. The delay cost of request a_j is given by $d_j = (g_i - a_j)$. The delay cost of bundle i, *i.e.*, $D_A(X_i)$, can be any arbitrary function of request delays d_j, for all $a_j \in X_i$. Two commonly used delay cost functions are *cumulative delay* and *max delay* functions [76]. Under the cumulative delay function,

$$D_A(X_i) = \sum_{a_j \in X_i} (g_i - a_j), \tag{4.3}$$

and under the max delay function,

$$D_A(X_i) = \max_{a_j \in X_i} (g_i - a_j)\,. \tag{4.4}$$

Notice that an equivalent form of the max delay function is given by $D_A(X_i) = g_i - a_{first,i}$, where $a_{first,i}$ is the arrival time of the first request in bundle i. It will become clear later, in Section 4.3, that the decision making technique used by our online algorithm is independent of the specific delay cost function considered and can be applied to a variety of other delay cost definitions. We will later comment on the effect of the specific choice of the delay cost function on the performance of our algorithm.

4.2 Optimal Offline Algorithm

The optimal offline algorithm knows the entire request arrival sequence a priori. While unrealistic, OPT can be used as a benchmark when evaluating the performance of our online algorithm. We design OPT based on the observation that every grant of the optimal algorithm happens right at the time of some request arrival, *i.e.*, for all $g_i \in \mathcal{G}_{\text{OPT}}$, we have $g_i = a_k$, for some $a_k \in \mathcal{A}$. That is, the optimal algorithm never makes a grant in-between two arrivals because doing so would only increase its cost. As a result, we can consider the problem of finding the optimal grant sequence as a discrete time optimization problem. Thus, upon each request arrival, the optimal algorithm should decide whether to make a grant for the current bundle or wait for the next request arrival.

Algorithm 1 OPT: Optimal Offline Algorithm

Input: $\mathcal{A} = \langle a_1, a_2, \ldots, a_n \rangle$
Output: C_{min}, Seq
1: **Initialize:** $C_{min}[1] = 0$
2: **Initialize:** $Seq[1] = \langle 1 \rangle$
3: **for** $i \in [2, n]$ **do**
4: $\quad C_{min}[i] = \alpha f_{delay}(1, i)$
5: $\quad Seq[i] = \mathcal{I}\langle i \rangle$
6: $\quad$ **for** $j \in [1, i-1]$ **do**
7: $\quad\quad \mathcal{C} = C_{min}[i-j] + \alpha f_{delay}(i-j+1, i)$
8: $\quad\quad\quad + \varepsilon(a_i - a_{i-j})$
9: $\quad\quad$ **if** $\mathcal{C} < C_{min}[i]$ **then**
10: $\quad\quad\quad C_{min}[i] = \mathcal{C}$
11: $\quad\quad\quad Seq[i] = \langle Seq[i-j], \mathcal{I}\langle j \rangle \rangle$
12: $\quad\quad$ **end if**
13: $\quad$ **end for**
14: **end for**

Specifically, we present a dynamic programming solution with the runtime of $O(n^2)$, where n is the length of the arrival sequence $\mathcal{A}$. The input to the algorithm is the arrival sequence $\mathcal{A}$. The output of the algorithm are arrays C_{min} and Seq. Here, $C_{min}[i]$ represents the optimal cost of the subsequence $\mathcal{A}_i = \langle a_1, \ldots, a_i \rangle$, and $Seq[i]$ is a *binary grant sequence* representing the decisions of OPT for the subsequence $\mathcal{A}_i$. If $Seq[i][j] = 1$, then assuming $\mathcal{A}_i$ as input, OPT makes a grant at a_j, otherwise it waits for the next arrival. Also, C_{OPT} is

given by $C_{min}[n] + T$.

Algorithm 1 shows the steps in OPT. As can be seen, it constructs the optimal grant sequence in a bottom-up fashion. The main idea of the algorithm is that, for any arrival sequence, there is a grant right when the last request arrives, and the second to last grant can happen when any of the previous requests arrives. Specifically, OPT consists of a nested loop where each iteration of the outer loop finds the optimal solution for the smaller subsequence $\mathcal{A}_i$ and stores it in $C_{min}[i]$ and $Seq[i]$. Finally, the last iteration of the outer loop computes the optimal grant decisions for the arrival sequence $\mathcal{A} = \mathcal{A}_n$ using the results achieved in previous iterations. The notation $f_{delay}(j, i)$, for $j \leq i$, denotes the delay cost incurred by requests $\langle a_j, \ldots, a_i \rangle$ when they are granted at time a_i. In the case of max delay function, we have $f_{delay}(j, i) = a_i - a_j$, and in the case of cumulative delay function we have $f_{delay}(j, i) = \sum_{k=j}^{i}(a_i - a_k)$. Also, $\mathcal{I}\langle l \rangle$ denotes a binary sequence of length l in which all elements are 0 except the last one, which is set to 1.

4.2.1 Performance of the Offline Algorithm

For the arrival sequence $\mathcal{A}$ (of length n), the last arrival will always trigger a grant. As a result, in order to find the optimal grant sequence for $\mathcal{A}$, there are 2^{n-1} possible binary grant sequences that should be considered. However, algorithm OPT is capable of finding such an optimal sequence in $O(n^2)$ time. This stems from the bottom up nature of the OPT. In the following, we discuss how OPT can attain the optimal solution in polynomial time, despite the existence of an exponential number of possible solutions for the problem.

Given the trivial solution for an arrival sequence of size 1, OPT starts by finding the optimal grant sequence for $\mathcal{A}_2$ (see Fig. 4.3). To do so, it compares the cost of two grant sequences $\langle 0, 1 \rangle$ and $\langle 1, 1 \rangle$. For the sake of argument, assume that $\langle 0, 1 \rangle$ results in a lower cost and hence OPT assigns it to $Seq[2]$. Thus, the optimal grant sequence for $\mathcal{A}$ can never start with the sequence $\langle 1, 1 \rangle$ which means that all the $2^{n-1-2} = 2^{n-3}$ grant sequences that start with $\langle 1, 1 \rangle$ are eliminated. Assuming $Seq[2] = \langle 0, 1 \rangle$, in the next step, OPT considers

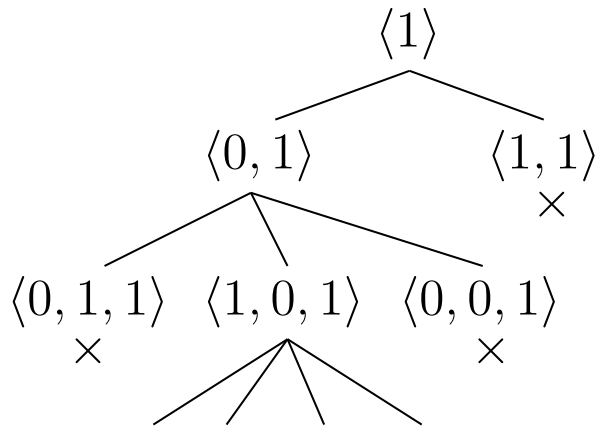


Figure 4.3: Bottom up nature of the offline algorithm.

3 solutions for $\mathcal{A}_3$, namely $\langle 1,0,1\rangle$, $\langle 0,1,1\rangle$, and $\langle 0,0,1\rangle$. Again one of these 3 cases will result in the lowest cost (assume that $Seq[3] = \langle 1,0,1\rangle$). This implies that the optimal grant sequence for $\mathcal{A}$, can never start with any of the other two sequences, *i.e.*, $\langle 0,1,1\rangle$ and $\langle 0,0,1\rangle$, thus eliminating $2 \times 2^{n-1-3} = 2^{n-3}$ of the grant sequences that start with them. Notice that the sequences eliminated in this step are different than the ones eliminated during the previous step (associated with $\mathcal{A}_2$). By following a similar routine, we can see that the bottom up nature of the OPT can eliminate an exponential number of the possible grant sequences for the arrival sequence $\mathcal{A}$.

Theorem 1. *When* $\alpha \geq 1$, OPT *makes a grant for every request arrival.*

Proof. We prove this Theorem by contradiction. Assume that there exists an arrival subsequence $\mathcal{A}_{jk} = \langle a_j, \ldots, a_k\rangle$ with $j < k$, where OPT bundles requests in this subsequence and makes a single grant at a_k (with a grant called g_{i+1}) instead of making individual grants (see Fig. 4.4). Also, assume that $\mathcal{A}_{jk}$ is extended to the maximum possible length in a sense that the last arrival before a_j is associated with a grant (we call this grant g_i). Let g_{i+2} denote the first grant of OPT after $\mathcal{A}_{jk}$. If we modify the optimal grant sequence by adding individual grants for every arrival in $\mathcal{A}_{jk}$, the weighted latency cost of the new sequence decreases by at least $\alpha(a_k - a_j)$. Rewriting this value using telescoping sums, we can quantify the decrease

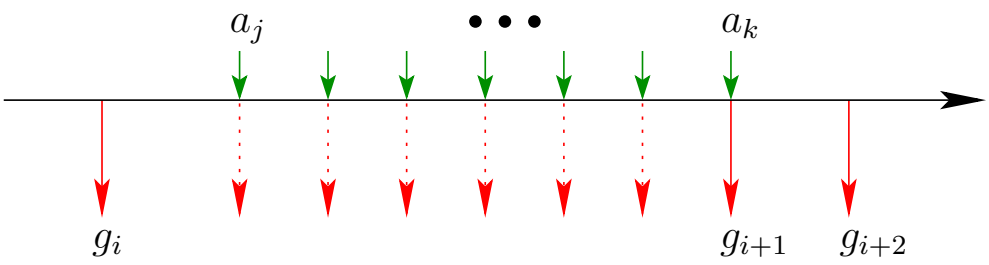


Figure 4.4: Modified grant sequence in Theorem 1.

in weighted latency cost as follows:

$$\alpha \sum_{l=j+1}^{l=k} a_l - a_{l-1} \,.$$

On the other hand, adding these new grants changes the energy cost contribution of $\mathcal{A}_{jk}$ from $\varepsilon(a_k - g_i)$ to $\varepsilon(a_j - g_i) + \sum_{l=j+1}^{l=k} \varepsilon(a_l - a_{l-1})$. Put another way, the net increase in the energy cost can be characterized as follows:

$$\varepsilon(a_j - g_i) - \varepsilon(a_k - g_i) + \sum_{l=j+1}^{l=k} \varepsilon(a_l - a_{l-1}) \,.$$

Notice that when $\alpha > 1$, we have $\alpha\tau > \varepsilon(\tau)$. In addition, $\varepsilon(a_j - g_i) - \varepsilon(a_k - g_i) \leq 0$ due to the fact that $\varepsilon(t)$ is a non-decreasing function. As a result, we can see that the increase in energy cost is less than the decrease in weighted latency cost. Thus, the cost of the modified grant sequence is less than the cost of the optimal grant sequence, which is a contradiction.

Interestingly, using a similar argument, it can be established that when $\alpha = 1$, there exist multiple optimal grant sequences and that granting at each arrival is one of them. ■

Notice that Theorem 1 and its proof are independent of any specific delay function and apply to both cumulative and max delay functions.

4.3 Online Break-Even Algorithm

With Theorem 1 characterizing the behavior of the online algorithm for $\alpha \geq 1$, our online algorithm called Break-Even (BE) will imitate OPT in that regime. In other regimes, BE

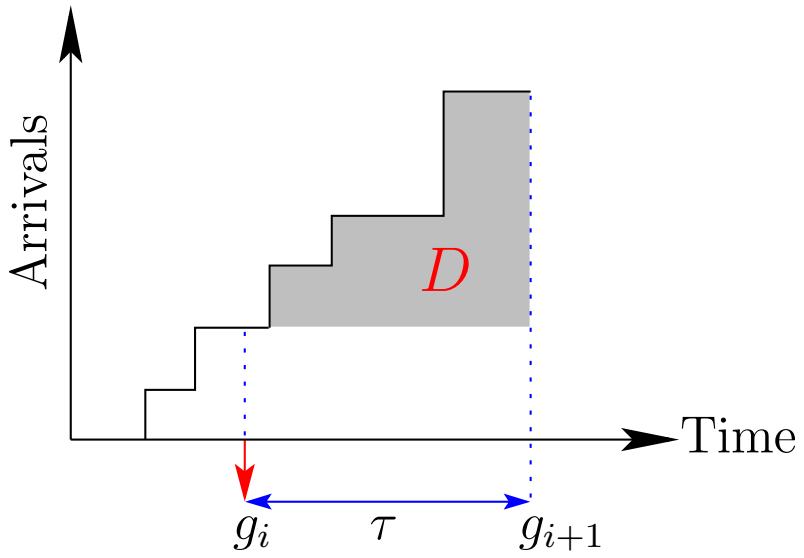


Figure 4.5: An example depicting the cumulative delay cost.

works as follows. Assume that the last grant was at time g_i and the algorithm has to decide when to make its next grant g_{i+1}. Let τ denote the time duration since the last grant g_i (see Fig. 4.2). Also, let $E_{\text{BE}}(\tau)$ and $D_{\text{BE}}(\tau)$ denote the energy and delay cost incurred by BE during τ. Then, BE makes a grant at time $g_{i+1} = g_i + \tau$, when $E_{\text{BE}}(\tau) = \alpha D_{\text{BE}}(\tau)$ holds. The first grant is treated differently, *i.e.*, when there is no g_i. The algorithm makes its first grant at some time τ that satisfies the equation $T = \alpha D_{\text{BE}}(\tau)$. The algorithm BE can be implemented using timers. In the following, we present the specific details of the algorithm and its implementation for each of the cumulative and max delay functions.

4.3.1 Cumulative Delay

In Fig. 4.5, the staircase curve shows the arrival of requests over time. It is easy to see that for any choice of τ, the area of the resulted shaded region will be equal to the cumulative delay cost associated with a grant at time $g_{i+1} = g_i + \tau$. Let D denote the numerical value of this area. BE will choose such a τ value that satisfies the following equation:

$$\min\{\tau, T\} = \alpha D\,. \tag{4.5}$$

To implement BE, upon the arrival of the l-th request at time a_l, a timer is set to time

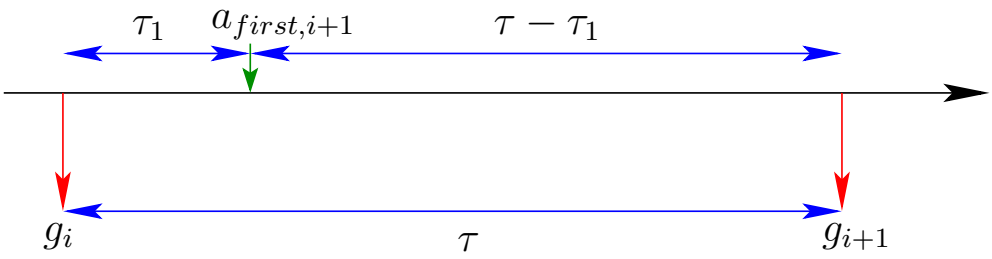


Figure 4.6: An example grant interval created by BE under max delay.

out at time $a_l+\tau$, where the value of τ is computed based on (4.5). If the next request arrives before the expiry of this timer, the timer value will reset to a new value τ by solving (4.5) with a new value of D. Otherwise, a time out at time $g_{i+1} = a_l + \tau$ will provoke a grant event in which case all the pending requests will be granted.

4.3.2 Max Delay

In Fig. 4.6, τ_1 is the time between the last grant of the algorithm (*i.e.*, g_i) and the first arrival after that. This way, the weighted delay cost associated with the grant at $g_i + \tau$ will be $\alpha(\tau - \tau_1)$. BE will choose such a τ value that satisfies the following equation:

$$\min\{\tau, T\} = \alpha(\tau - \tau_1)\,. \tag{4.6}$$

Since $\alpha < 1$, the only feasible solution for (4.6) is $\tau - \tau_1 = \frac{T}{\alpha}$. Therefore, for the first arrival after each grant g_i, BE sets a timer to time out after $\frac{T}{\alpha}$ time units. Upon expiry of the timer, a grant will be made and all the pending requests will be granted.

4.4 Analysis of Online Algorithm under Cumulative Delay

As mentioned earlier, in the regime of $\alpha \geq 1$, BE imitates the behavior of OPT as determined by Theorem 1. As such, BE is 1-competitive when $\alpha \geq 1$. Therefore, it suffices to analyze BE for the regime of $\alpha < 1$. In this regime, the main difficulty in determining the competitive ratio of BE is that the algorithm can make several grants in a short period of time and incur only a small energy cost. This implies that, an algorithm can make many grants and incur

smaller energy cost compared to another algorithm that makes only a few grants. This is completely different from the problem setting considered in [16, 74, 76], where it is assumed that if an algorithm makes more grants then it incurs a higher energy cost.

In the reminder of this section, we focus on proving the following theorems.

Theorem 2. *For cumulative delay function, the Break-Even algorithm achieves a competitive ratio of* 4*, that is,* $C_{\mathrm{BE}} \leq 4\, C_{\mathrm{OPT}}$.

Theorem 3. *For cumulative delay function, the competitive ratio of* 4 *is tight, that is, there is an arrival sequence for which* $C_{\mathrm{BE}} = 4\, C_{\mathrm{OPT}}$.

4.4.1 Preliminaries

The main idea is to lower-bound the cost of OPT with respect to the cost of BE.

Lemma 1. *The cost of grant interval* X_i *is given by* $C_{\mathrm{BE}}(X_i) = 2 \min\{X_i, T\}$.

Proof. Recall that BE makes a grant when $E_{\mathrm{BE}}(X_i) = \alpha D_{\mathrm{BE}}(X_i)$. Thus, the cost of interval i is given by $C_{\mathrm{BE}}(X_i) = 2E_{\mathrm{BE}}(X_i)$. Based on the definition of the energy cost, we have $E_{\mathrm{BE}}(X_i) = \min\{X_i, T\}$. This completes the proof. ∎

Observation 1. *At least one request arrives during an interval* $X_i = (g_{i-1}, g_i]$. *If there is no arrival, then the algorithm does not make any grant at* g_i *as there is no request to grant.*

Observation 2. *Using Lemma 1, the cost of interval* X_i *satisfies the relations* $C_{\mathrm{BE}}(X_i) \leq 2X_i$ *and* $C_{\mathrm{BE}}(X_i) \leq 2T$.

4.4.2 Radio State Transitions

Let $\mathcal{G}_{OPT} = \langle g_1, \ldots, g_l \rangle$ denote the sequence of grants made by OPT. Recall that OPT knows the sequence $\mathcal{A}$ in advance, and hence can optimally space its grants in order to minimize its cost. *We make no assumption about the relation between the number of grants of* BE *and* OPT.

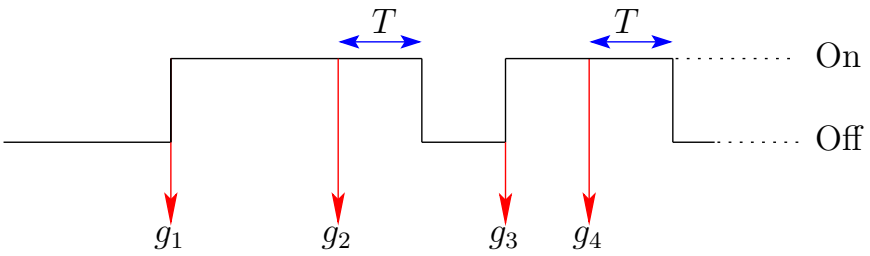


Figure 4.7: Grants and radio state transitions of OPT.

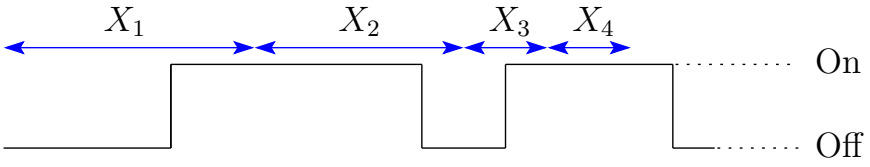


Figure 4.8: Grant intervals of BE overlaid on radio states of OPT.

Fig. 4.7 depicts the grants of OPT and the corresponding state of the radio interface under this algorithm. At the beginning, the radio is off. During the execution of the algorithm, the radio may switch between the on and off states several times. Once the last grant is made, which happens when the last request arrives at time a_n, after a tail time T, the radio goes to the off state again.

Fig. 4.8 depicts the grant intervals of BE overlaid on the radio states under OPT. Two types of intervals can be identified:

1. Intervals that do not include any radio state transitions.

2. Intervals that include one or more radio state transitions.

The following subsections, analyze these two types of interval.

4.4.3 Intervals with No Radio Transition

Let X denote such an interval. There are two types of such intervals:

1. *Radio is on during interval X:* In this case, no matter how many grants OPT makes, it incurs at least the energy cost as its radio is on. Its delay cost could be zero, but it keeps

the radio on for at least X amount of time. Thus,

$$C_{\mathrm{OPT}}(X) \geq X\,. \tag{4.7}$$

Based on Observation 2, it is obtained that,

$$C_{\mathrm{BE}}(X) = 2\min\{X, T\} \leq 2X \leq 2C_{\mathrm{OPT}}(X)\,. \tag{4.8}$$

2. *Radio is off during interval X:* Following Observation 1, since BE makes a grant at the end of interval X, it means that at least one request has arrived during X. As the radio is off for OPT, we conclude that OPT incurs at least a delay cost equal to $D_{\mathrm{BE}}(X)$, as does BE. Therefore,

$$C_{\mathrm{OPT}}(X) \geq \alpha D_{\mathrm{BE}}(X)\,. \tag{4.9}$$

As discussed in the proof of Lemma 1, we have $C_{\mathrm{BE}}(X) = 2E_{\mathrm{BE}}(X) = 2\alpha D_{\mathrm{BE}}(X)$. Thus, the following relation is obtained,

$$C_{\mathrm{BE}}(X) = 2\alpha D_{\mathrm{BE}}(X) \leq 2C_{\mathrm{OPT}}(X)\,. \tag{4.10}$$

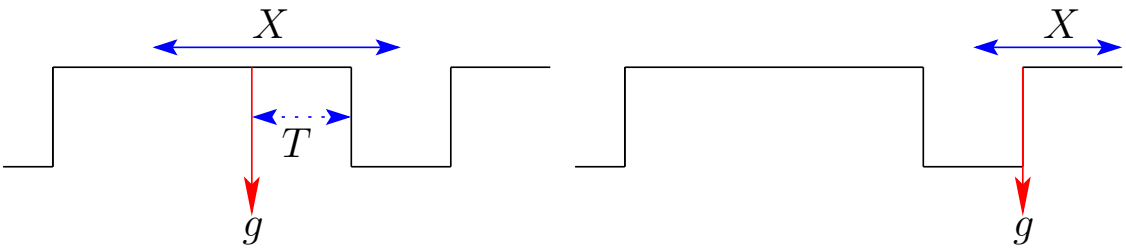
4.4.4 Intervals with One or More Radio Transitions

Let X denote such an interval. Consider the grants of OPT during the interval X. If there are no grants, then clearly $C_{\mathrm{BE}}(X) \leq 2C_{\mathrm{OPT}}(X)$ as OPT suffers from at least a delay cost equal to $D_{\mathrm{BE}}(X)$ because it does not make any grants during X. Thus, in the remainder of this subsection, we consider the case that OPT makes at least one grant during interval X.

Consider the first grant of OPT in X. As depicted in Fig. 4.9, there are two cases to be considered:

1. *The first grant happens when the radio is already on:* This case is depicted in Fig. 4.9(a). Since OPT has a grant when the radio is on, the radio remains on for at least a tail time T. Thus, $C_{\mathrm{OPT}}(X) \geq T$. It then follows that,

$$C_{\mathrm{BE}}(X) \leq 2\min\{X, T\} \leq 2T \leq 2C_{\mathrm{OPT}}(X)\,. \tag{4.11}$$



(a) First grant of OPT happens when the radio is on. (b) First grant of OPT happens when the radio is off.

Figure 4.9: Intervals with one or more radio transitions.

2. *The first grant happens when the radio is off:* This case is depicted in Fig. 4.9(b). In this case, the cost of OPT could be as low as zero if its first grant (denoted by g on the figure) happens right at the same time as the grant of X. Thus, the only relation that can be established in this case is,

$$C_{\text{BE}}(X) \leq 2 \min\{X, T\} \leq 2T, \tag{4.12}$$

and the ratio $C_{\text{BE}}(X)/C_{\text{OPT}}(X)$ is indeed unbounded. Let $\mathcal{X}^0_{\text{BE}} \subseteq \mathcal{X}_{\text{BE}}$ denote the set of all such intervals of BE.

The key idea is then to bound the cost of OPT over the interval set $\mathcal{X}^0_{\text{BE}}$ rather than bounding its cost over individual intervals (where its cost could be zero). This is established during the proof of Theorem 2 presented next.

4.4.5 Proof of Theorem 2

Proof. We have the following relations for the cost of BE and OPT with respect to grant intervals $\mathcal{X}_{\text{BE}}$,

$$C_{\text{OPT}} = \sum_{X_i \in \mathcal{X}_{\text{BE}}} C_{\text{OPT}}(X_i) + T, \tag{4.13}$$

$$C_{\text{BE}} = \sum_{X_i \in \mathcal{X}_{\text{BE}}} C_{\text{BE}}(X_i) + T\,. \tag{4.14}$$

It was shown in the previous subsections that for every interval $X_i \in \mathcal{X}_{\text{BE}}/\mathcal{X}^0_{\text{BE}}$, we have, $C_{\text{BE}}(X_i) \leq 2C_{\text{OPT}}(X_i)$. Also, for every interval $X_i \in \mathcal{X}^0_{\text{BE}}$, we have, $C_{\text{BE}}(X_i) \leq 2T$. There-

fore, it is obtained that,

$$\begin{aligned} C_{\mathrm{BE}} &= \sum_{X_i \in \mathcal{X}_{\mathrm{BE}}} C_{\mathrm{BE}}(X_i) + T \\ &= \sum_{X_i \in \mathcal{X}_{\mathrm{BE}} \setminus \mathcal{X}^0_{\mathrm{BE}}} C_{\mathrm{BE}}(X_i) + \sum_{X_i \in \mathcal{X}^0_{\mathrm{BE}}} C_{\mathrm{BE}}(X_i) + T \\ &\leq 2 \sum_{X_i \in \mathcal{X}_{\mathrm{BE}} \setminus \mathcal{X}^0_{\mathrm{BE}}} C_{\mathrm{OPT}}(X_i) + \sum_{X_i \in \mathcal{X}^0_{\mathrm{BE}}} (2T) + T \\ &\leq 2C_{\mathrm{OPT}} + 2T\,|\mathcal{X}^0_{\mathrm{BE}}|, \end{aligned} \tag{4.15}$$

where $|\mathcal{X}^0_{\mathrm{BE}}|$ denotes the cardinality of set $\mathcal{X}^0_{\mathrm{BE}}$. Thus, it is left to compute an upper bound for the term $|\mathcal{X}^0_{\mathrm{BE}}|$. To this end, we focus on the behavior of OPT and observe that for the entire arrival sequence, the OPT radio will transition between different states several times. Assume that for $\mathcal{K}$ times, there is a transition from the off to on state. Accordingly, we have $C_{\mathrm{OPT}} \geq \mathcal{K}T$, because every time the radio goes to the on state, it incurs at least an energy cost equal to one tail time T before going back to the off state. Also notice that after overlaying BE intervals over the radio states under OPT, we cannot have more than $\mathcal{K}$ intervals belonging to $\mathcal{X}^0_{\mathrm{BE}}$. Thus, it is obtained that,

$$|\mathcal{X}^0_{\mathrm{BE}}| \leq \mathcal{K} \leq \frac{C_{\mathrm{OPT}}}{T}\,. \tag{4.16}$$

By replacing $|\mathcal{X}^0_{\mathrm{BE}}|$ in (4.15) with its upper bound from (4.16), the following relation is obtained,

$$\begin{aligned} C_{\mathrm{BE}} &\leq 2C_{\mathrm{OPT}} + 2T|\mathcal{X}^0_{\mathrm{BE}}| \\ &\leq 2C_{\mathrm{OPT}} + 2C_{\mathrm{OPT}} = 4C_{\mathrm{OPT}}\,. \end{aligned} \tag{4.17}$$

■

4.4.6 Proof of Theorem 3

Proof. To show that the competitive ratio 4 is tight, it is sufficient to provide an example that attains this ratio. To this end, consider the scenario depicted in Fig. 4.10. Assume that $\alpha < 1$, *i.e.*, energy is more important than delay.

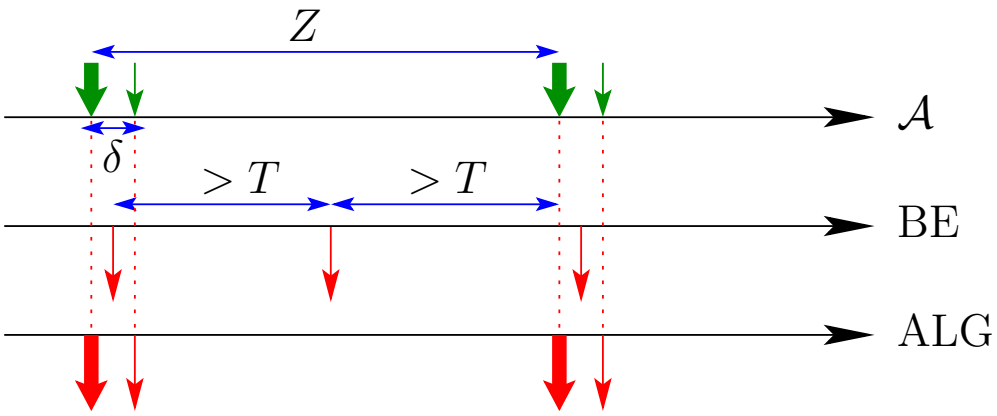


Figure 4.10: Example for the lower bound under cumulative delay.

In this example, every time the radio is off, a large number of requests (depicted by a thick arrow) arrive in a short period of time, *i.e.*, a batch arrival, so that the online algorithm makes a grant immediately after the batch arrival. Recall that individual request delays are added together. Therefore, even though individual delays are small, a large number of them are added together to become equal to T, at which point BE makes its first grant. Since the radio is off, the cost of the first grant is given by its delay cost, which is equal to T.

A short time δ after this grant, another request arrives (depicted by a thin arrow) and then nothing arrives for a while. Algorithm BE waits for some time before making a grant for the second arrival. Since $\alpha < 1$, the wait time will be longer than T in order to satisfy the equation $E_{\mathrm{BE}}(\delta + t) = \min\{\delta + t, T\} = \alpha t$. For the second grant, the cost is the summation of the energy cost T and its weighted delay cost, which is also equal to T, for a total cost of $2T$. Once time T has passed after the second grant, the radio goes to the off state, which incurs the energy cost T due to the tail time. At this time, the same scenario is repeated again (*i.e.*, a large batch arrival immediately followed by a single arrival).

Next, consider the cost of OPT for the same scenario. We do not know how OPT behaves in this case, but we do know that *its cost is less than or equal to the cost of the algorithm that makes a grant as soon as any request arrives.* This algorithm is denoted by ALG on Fig.4.10. For this algorithm, its delay cost will be zero, and thus its total cost is given by

the radio on time δ plus a tail time.

Let Z denote the time interval from when the first request in a batch arrives until the arrival of the first request of the next batch as depicted in Fig.4.10. Based on the above argument, it is obtained that,

$$C_{\mathrm{BE}}(Z) = T + 2T + T, \tag{4.18}$$

$$C_{\mathrm{OPT}}(Z) \leq C_{\mathrm{ALG}}(Z) = \delta + T\,. \tag{4.19}$$

The proof is established by noting that,

$$\lim_{\delta \to 0} \frac{C_{\mathrm{BE}}}{C_{\mathrm{OPT}}} = 4\,. \tag{4.20}$$

■

4.5 Analysis of Online Algorithm under Max Delay

As stated before, when $\alpha \geq 1$, BE follows the behavior of OPT specified by Theorem 1. In other words, BE is 1-competitive when $\alpha \geq 1$. In the following, we prove that under max delay function and in the regime of $\alpha < 1$, BE achieves a competitive ratio of 2. We also prove that the competitive ratio of 2 is tight.

4.5.1 2-competitiveness

Under max delay function and in the regime of $\alpha < 1$, BE divides time horizon into a set of intervals, each of length at least $\frac{T}{\alpha}$. Fig. 4.11 shows one such interval, which we refer to as X. Considering the distance of $\frac{T}{\alpha}$ between the first arrival in X and its associated grant, the weighted delay cost incurred by BE will be $\alpha \times \frac{T}{\alpha} = T$. Since BE makes a grant when $E_{\mathrm{BE}}(X) = \alpha D_{\mathrm{BE}}(X)$, the cost of interval X is

$$C_{\mathrm{BE}}(X) = 2\alpha D_{\mathrm{BE}}(X) = 2E_{\mathrm{BE}}(X) = 2T\,. \tag{4.21}$$

Depending on whether OPT has a grant in X, we can consider two types of intervals:

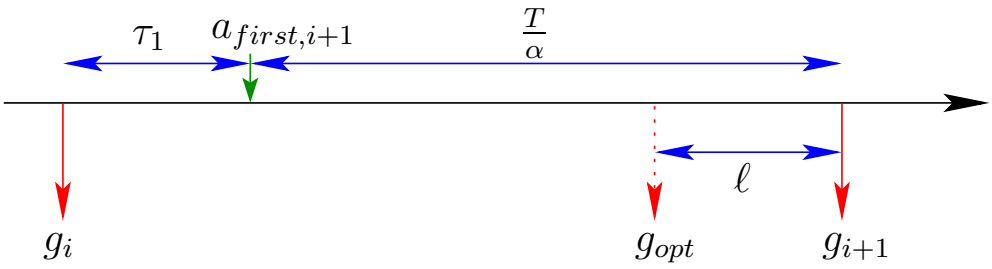


Figure 4.11: $(g_i, g_{i+1}]$ is a sample interval created by BE under max delay.

1. Intervals with no grant from OPT

2. Intervals with at least one grant from OPT

4.5.1.1 Intervals with no grant from OPT

If OPT does not make any grant in X, it will incur at least a delay cost equal to $D_{\text{BE}}(X)$, and hence $C_{\text{OPT}}(X) \geq \alpha D_{\text{BE}}(Y)$, which establishes $C_{\text{BE}}(X) \leq 2C_{\text{OPT}}(X)$.

4.5.1.2 Intervals with at least one grant from OPT

In case OPT has at least one grant in X, we will use g_{OPT} to refer to its first grant in this interval. Let ℓ denote the time duration from g_{OPT} up to the end of interval X (see Fig. 4.11). We will charge g_{OPT} with $\varepsilon(\ell)$ for its contribution to the energy cost and $\alpha(\frac{T}{\alpha} - \ell)$ for its contribution to the weighted delay cost. If $\ell > T$, then $C_{\text{OPT}}(X) \geq \varepsilon(\ell) = T$. Since $T \geq E_{\text{BE}}(X)$, we obtain that $2C_{\text{OPT}}(X) \geq C_{\text{BE}}(X)$. Otherwise (*i.e.*, if $\ell \leq T$), we have,

$$\begin{aligned} C_{\text{OPT}}(X) &\geq \alpha(\frac{T}{\alpha} - \ell) + \varepsilon(\ell) \\ &= T + (1-\alpha)\ell\,. \end{aligned} \tag{4.22}$$

$C_{\text{OPT}}(X)$ is lower bounded in (4.22) because it is possible for OPT to have more than one grant in interval X. When $\alpha < 1$ in (4.22), ℓ's coefficient becomes positive. As a result, it always holds that $T \leq C_{\text{OPT}}(X)$, which implies that $C_{\text{BE}}(X) \leq 2C_{\text{OPT}}(X)$.

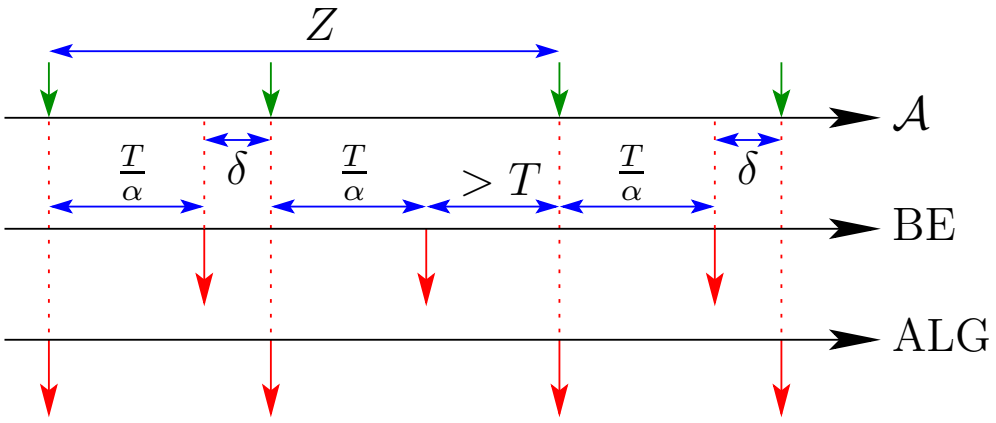


Figure 4.12: Example for the lower bound under max delay.

4.5.2 Tightness

Similar to the proof of Theorem 3, it suffices to provide an example arrival sequence that attains competitive ratio of 2. To this end, consider the arrival sequence in Fig. 4.12 when $\alpha < 1$.

In Fig. 4.12, every time the radio is off, a single request arrives. As mentioned in section 4.3, BE waits until $\alpha D_{\text{BE}}(\tau) = T$ holds, meaning that it makes a grant T/α time units after the arrival of this request. Since the radio was off, the grant cost consists of only the weighted delay cost which is T. Some time δ after this grant, another request arrives, and again based on the policy of $\alpha D_{\text{BE}}(\tau) = E_{\text{BE}}(\tau)$, BE makes a grant T/α time units after this arrival. The cost of the second grant consists of both the energy cost and the weighted delay cost. Each of these components are equal to T, adding up to the total cost of $2T$. After the second grant, no request arrives for a duration of at least T. During this period, the radio returns to the off state and energy cost of T is incurred due to the tail time. From this point onward, the same arrival pattern is repeated again.

We now consider the cost of the algorithm ALG that grants requests as soon as they arrive. Over the time interval Z (specified in Fig. 4.12), ALG has a delay cost of zero and its total cost is $2T$, which is the summation of the tail times associated with the first two

grants. Given that C_{OPT} is upper bounded by C_{ALG}, we have the following relations,

$$\begin{aligned} C_{\mathrm{BE}}(Z) &= T + 2T + T = 4T, \\ C_{\mathrm{OPT}}(Z) &\leq C_{\mathrm{ALG}}(Z) = 2T\,. \end{aligned} \tag{4.23}$$

The relations in 4.23 establish a lower bound of 2 on the competitive ratio of BE for max delay function. On the other hand, we know from the competitive analysis presented earlier that the competitive ratio is upper bounded by 2. This implies that the competitive ratio of the example arrival sequence in Fig. 4.12 is 2.

4.6 Performance Evaluation

In this section, we first present model-driven simulation results to verify the accuracy of our results. Then, we present experimental results based on measurements on a smartphone to demonstrate the utility and performance of the proposed algorithm in realistic scenarios. In addition to BE and OPT, we have also implemented the *Default* algorithm, in which requests are granted as soon as they arrive. Unless indicated to the contrary, we only present the results for cumulative delay function in experiments where similar conclusions can be made about BE's behavior under both delay functions. Also, given that BE has the same behavior for the entire regime of $\alpha \geq 1$, we only present the experimental results for $\alpha = 1$ to avoid redundancy.

4.6.1 Model-Driven Evaluation

In this part, we use a custom-developed discrete-event simulator to compute energy and delay costs under different algorithms. The input to the simulator consists of the weight factor α, the value of the tail time and the arrival sequence. Unless otherwise stated, the tail time is set to 200 ms, which is the value for the tail time of the Continuous Reception substate in LTE's Radio Resource Control (RRC) state machine [14].

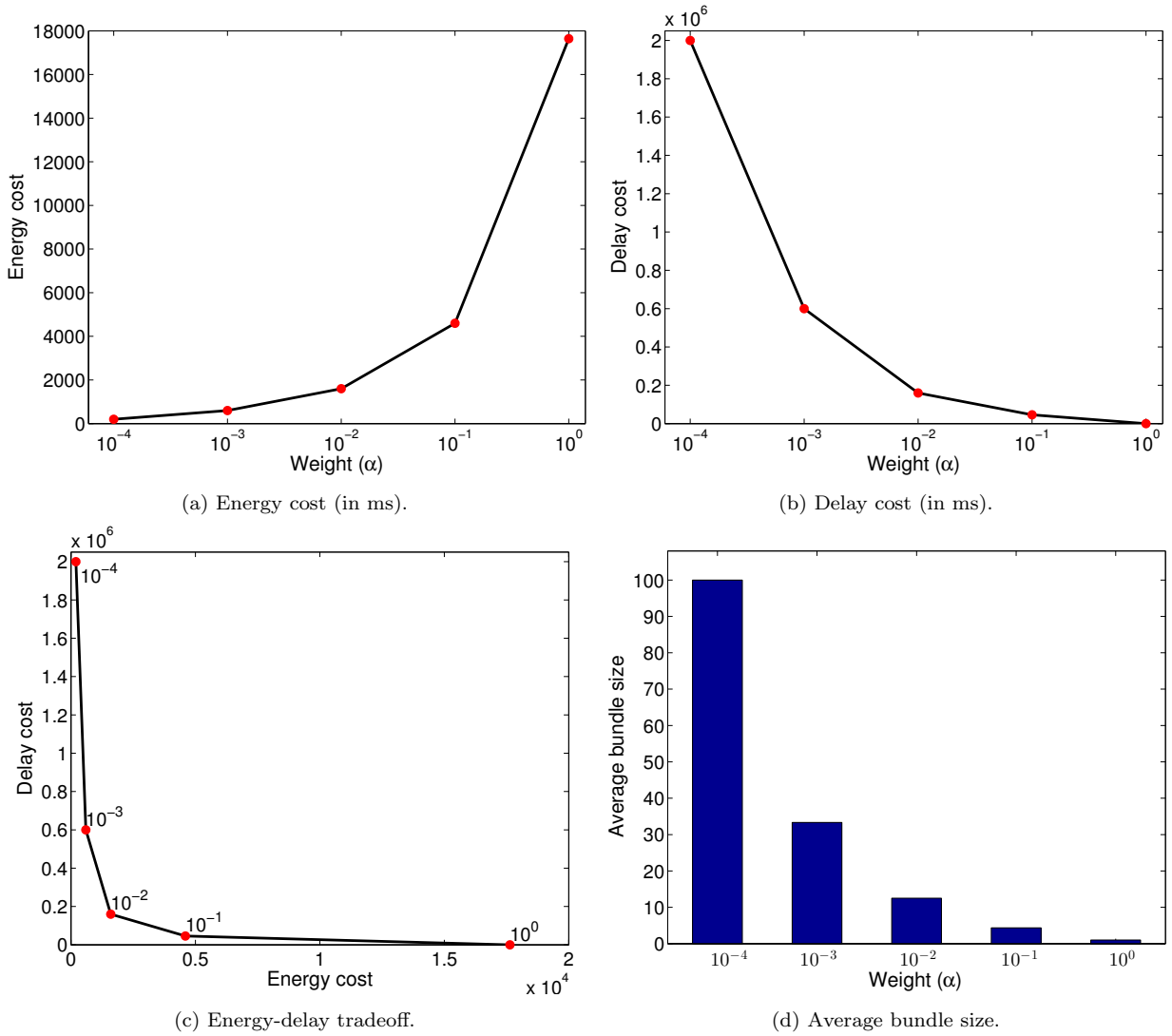
4.6.1.1 Exploring Energy-Delay Tradeoff

The first experiment is performed using a sequence of 100 request arrivals, where the inter-arrivals are sampled from a normal distribution with mean 200 ms and standard deviation 80 ms. In this sequence, about 41% (59%) of the inter-arrival times are less (greater) than the tail time.

Figs. 4.13(a) and 4.13(b) represent the energy and delay cost for different values of α, respectively. It can be seen that by exploring the large parameter space of α, BE is able to provide different levels of energy savings depending on the user preference. Specifically, by increasing the weight α, BE achieves lower delay values at the expense of higher energy consumption. In our experiments, $\alpha = 10^{-4}$ and $\alpha = 1$ mark two ends of the spectrum where maximum energy saving (and delay reduction) are achieved at the expense of increased delay (and energy consumption). For example, $\alpha = 1$ results in 100% delay reduction compared to $\alpha = 10^{-4}$, while $\alpha = 10^{-4}$ brings about 98.9% energy saving compared to $\alpha = 1$. Fig. 4.13(c) combines the previous two plots by showing the pairwise values of energy and delay along with their corresponding weight factors.

Fig. 4.13(d) shows the average size of the bundles created by BE. It is observed that BE ia able to adjust its behavior based on the weight given to the delay cost. Specifically, the average bundle size decreases from 100 to 1 by increasing the delay weight from 10^{-4} to 1. For lower values of α, greater energy savings require a more aggressive aggregation policy, thus creating larger bundles. On the other hand, in scenarios where delay has higher weight, BE tends to avoid bundling and grants requests as soon as they arrive.

Using the same arrival sequence, we also run the BE algorithm under the max delay function. In order to compare the performance of BE under different delay functions, we measured the delay experienced by individual requests when using each delay function. Fig. 4.14 presents the cumulative distribution function (CDF) of request delays under cumulative and max delay functions for two values of α. It can be seen that the cumulative function results



(a) Energy cost (in ms).

(b) Delay cost (in ms).

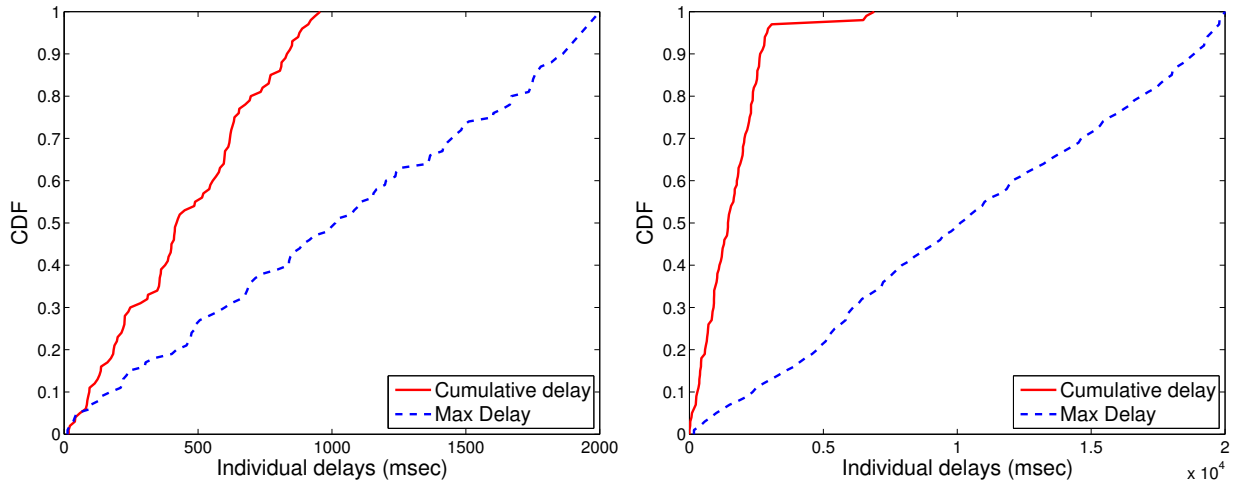
(c) Energy-delay tradeoff.

(d) Average bundle size.

Figure 4.13: Performance of BE: By controlling α, different energy-delay tradeoffs can be achieved.

in lower delays for requests. For example, in the case of $\alpha = 10^{-1}$, 50% of requests incur delays less than 420 ms when cumulative delay function is used. However, the median delay is about 1013 ms when using the max delay function. Fig. 4.14 also reveals that regardless of the delay function, changing the weight factor (α) affects the distribution of individual request delays.

Table 4.1 compares the performance of BE and OPT in terms of the empirical competitive ratio achieved for different values of α and different delay functions. It can be seen that under cumulative delay function, BE performs considerably better than the predicted worst-case



(a) $\alpha = 10^{-1}$ (b) $\alpha = 10^{-2}$

Figure 4.14: CDF of individual request delays under cumulative and max delay functions.

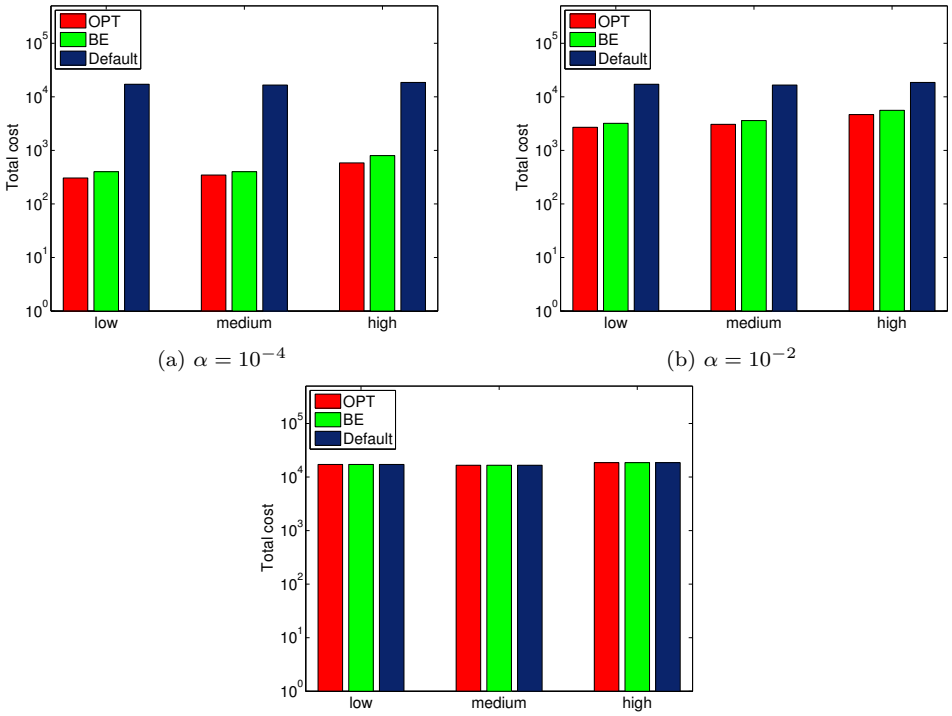
competitive ratio of 4. Also, under max delay function, the values of empirical competitive ratio conform with the 2-competitive result proved in our analysis. Notice that for $\alpha = 1$, the total cost of BE becomes equal to the cost of OPT. The reason is that in the regime of $\alpha \geq 1$, both BE and OPT have identical behaviors as they grant requests as they arrive.

Table 4.1: Empirical competitive ratio of BE.

	$C_{\text{BE}}/C_{\text{OPT}}$	
$\boldsymbol{\alpha}$	Cumulative delay	Max delay
10^{-4}	1.32	1.98
10^{-3}	1.30	1.81
10^{-2}	1.18	1.96
10^{-1}	1.17	1.75
1	1	1

4.6.1.2 Performance under Different Arrival Patterns

To study the behavior of BE under different arrival patterns, we consider the fluctuation level of the inter-arrival times. Similar to [101], we use the coefficient of variation (CV) to classify sequences of arrival times into groups of *low*, *medium* and *high* fluctuations. We conduct simulations with sequences characterized by CV = 0.5 (low fluctuation), CV = 1.5 (medium fluctuation) and CV = 5 (high fluctuation). In all sequences, inter-arrival times are sampled from a normal distribution with mean 200 ms.



(a) $\alpha = 10^{-4}$

(b) $\alpha = 10^{-2}$

(c) $\alpha = 1$

Figure 4.15: Comparing the performance of BE with OPT and Default under different fluctuation levels of request inter-arrival times.

Fig. 4.15 compares the total cost of the three algorithms under varying fluctuation levels and weight values. It is observed that for a specific delay weight, the performance of BE changes depending on the characteristics of the arrival sequence. For example in Fig. 4.15(a) ($\alpha = 10^{-4}$), the cost of BE is 1.15 and 1.37 times the cost of OPT for sequences with medium and high fluctuation, respectively. Among all the considered scenarios, the ratio $C_{\text{BE}}/C_{\text{OPT}}$ ranges from 1 to 1.37, which is consistent with our analysis. We can also see that, in scenarios with higher weight for energy ($\alpha = 10^{-2}, 10^{-4}$), BE outperforms the Default algorithm. For example, in a setting with $\alpha = 10^{-2}$ and high fluctuation, BE results in 69.8% reduction in the total cost compared to Default.

As Fig. 4.15(c) illustrates, in scenarios with high delay importance ($\alpha \geq 1$), all three algorithms have an identical performance as they grant requests as soon as they arrive. This is in line with the claim made by Theorem 1. Also as seen in Fig. 4.15, for a given input sequence, the total cost of the Default algorithm is independent of the weight factor. This is

due to the fact that the requests do not experience any delay under this algorithm and hence the total delay cost of the Default algorithm will always be zero. As a result, the weight assigned to the delay cost becomes ineffective under Default.

4.6.1.3 A Bursty Arrival Pattern

In this section, we verify that for cumulative delay function, the competitive ratio of BE is indeed greater than 2. To this end, we generate an arrival sequence inspired by the example described in Subsection 4.4.6. As Fig. 4.16 shows, this sequence is a repetition of a specific pattern, where a burst of requests (marked by a thick arrow) arrives and then after a short period of time, denoted by t_s, a single request arrives (marked by a thin arrow). Thereafter, a long interval, denoted by t_l, passes before the arrival of the next burst. By following this pattern, we construct a sequence of 500 requests, where the size of each burst is uniformly distributed between 1 and 14. The short and long time intervals (t_s and t_l) are exponentially distributed with means 40 ms and 400 ms, respectively.

Table 4.2: Empirical competitive ratio with bursty arrival pattern.

α	$C_{\text{BE}}/C_{\text{OPT}}$	α	$C_{\text{BE}}/C_{\text{OPT}}$
0.6	2.11	0.9	2.51
0.7	2.23	0.99	2.55
0.8	2.36	0.999	2.62

Table 4.2 tabulates the empirical competitive ratio of BE for values of α that result in a cost ratio greater than 2. We note that, while the cost ratio is greater than 2 in these scenarios, it is still consistent with the ratio 4.

4.6.1.4 Effect of Tail Time

To study the effect of the tail time on the performance of BE, we use a sequence of 100 requests, where the inter-arrival times are sampled from a normal distribution with mean 300 ms and standard deviation 80 ms. We perform a set of experiments with three different values for the tail time (100, 200 and 300 ms). Fig. 4.17 depicts the energy cost for different values of the tail time as a function of the delay weight (α). It is observed that

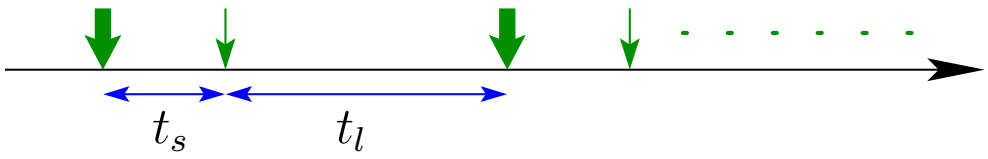


Figure 4.16: A bursty arrival sequence: The thick and thin arrows indicate burst and single arrivals, respectively

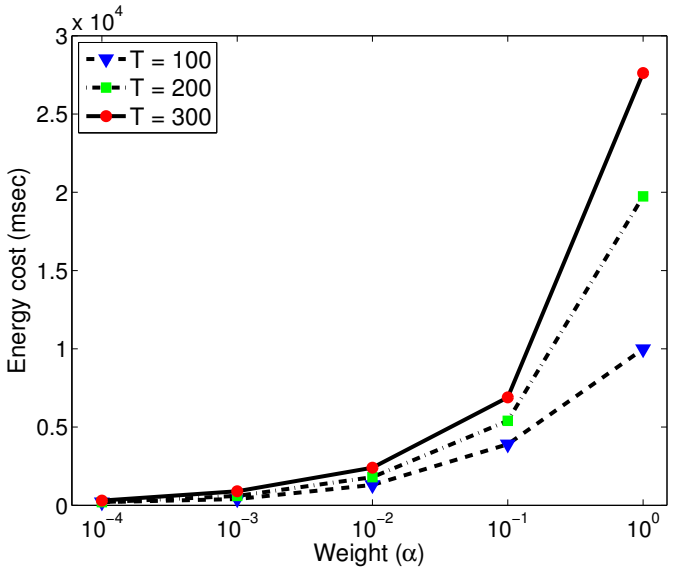
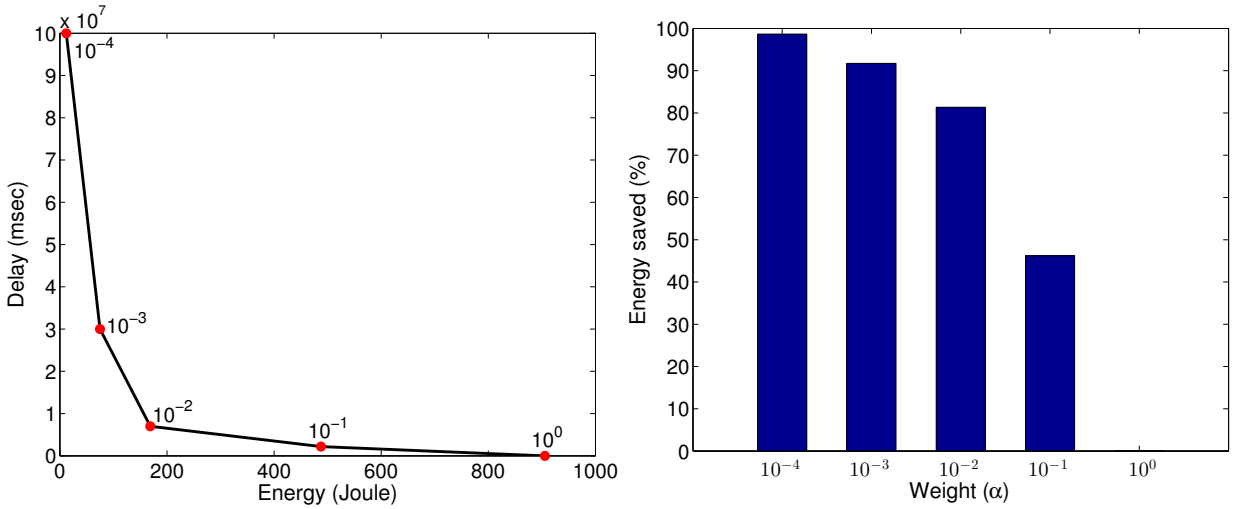


Figure 4.17: Energy cost of BE with different tail times.

the energy-delay tradeoff and hence the energy cost is dependent on the value of the tail time. Specifically, decreasing the tail time leads to a lower energy cost in almost all cases (α values). For example, in the case of $\alpha = 1$, an energy cost improvement of about 64% can be achieved when changing the tail time from 300 ms to 100 ms. This is due to the fact that larger values of tail time tend to keep the network interface on for longer, which can cause higher energy consumption.

Fig. 4.17 also shows that the benefit of shorter tail times grows with increasing the delay weight. The reason is that, to reduce delay, a higher number of grants are needed, which in turn generates a higher number of inter-grant idle gaps. Therefore, the role played by the tail time becomes more significant when delay has higher importance to the user. A similar argument can be made to justify the marginal benefits of shortening the tail time in settings



(a) Energy-delay tradeoff.

(b) Energy savings.

Figure 4.18: LTE experiments on a Nexus smartphone.

with higher energy weight ($\alpha \ll 1$).

4.6.2 Smartphone Experiments on LTE

To assess the performance of BE under more realistic conditions, we also performed experiments on a Nexus smartphone. To measure the energy consumption of the radio interface, we used the AT&T ARO tool [11], which is configured with AT&T LTE network parameters. To conduct experiments on a smartphone, we developed an Android app that turns off the screen and performs HTTP transfers at user-specified times. Each *run* of this app uses a grant sequence file and a URL as input. It then repeatedly downloads the object referred to by the URL at the times specified in the grant file. We also installed a firewall app (AFWall+) on the phone to block all background traffic from OS services and other apps.

For the input sequence, we created a sequence of 100 requests with normal inter-arrival times of mean 10 seconds and standard deviation 5 seconds, based on the measurement results reported in [13] for popular news feed updates. Also, given that the operation of BE depends on the value of the tail time, we used 10 seconds as the tail value which is the default value in ARO's AT&T LTE profile for the inactivity timer of the RRC_CONNECTED state.

Using the input sequence, and for different values of α, we run BE and Default and record their resulting grant times in separate files. We then feed those files to our Android app and measure the radio energy consumption of the device during each run.

Fig. 4.18(a) plots the pairwise energy-delay values of BE along with their corresponding α values. Notice that both Fig. 4.18(a) and Fig. 4.13(c) in previous section use the same delay measure which is the cumulative delay (in ms) incurred by all requests in the sequence. However, while Fig. 4.13(c) expresses energy in terms of the milliseconds spent in the high power state, here we present energy in terms of Joules. Fig. 4.18(a) illustrates a consistent behavior between simulation and real-world experiments as BE spans the broad spectrum of the energy-delay tradeoff. Also, our experiment with the Default algorithm (which only achieves a fixed tradeoff point) resulted in zero delay and an energy expenditure of 905.7 Joules. Fig. 4.18(b) plots the energy savings of BE compared to the Default algorithm for different values of α. We can see growing energy savings by increasing the relative importance of energy. Across all the values of α, the energy savings of BE can range between 0% and 99%.

Chapter 5

Online Energy Management in IoT Applications

In recent years, an ever increasing number of smart devices with sensing and communication capabilities has given rise to the Internet of Things (IoT). IoT envisions a world where a variety of smart objects are connected to the Internet and communicate with each other without human intervention. Gartner predicts that by 2020, about 35 billion smart objects will be connected to the Internet [102]. As a promising networking paradigm, IoT enables a new range of services such as smart homes [103], e-healthcare [104] and environmental monitoring [105].

With LTE being a natural candidate for IoT connectivity, in this chapter we consider energy management on LTE-enabled IoT devices. A characteristic feature of IoT applications is the periodic generation of small messages, whose transmission over LTE is highly energy inefficient. To mitigate the energy inefficiency resulting from small IoT messages, we resort to the message bundling technique. As mentioned in Chapter 1, the benefits of message bundling come at the expense of additional delay in message transmission, thus reflecting the existence of an energy-delay tradeoff. Therefore, we study a message bundling algorithm that is capable of balancing the enrgy-delay tradeoff and is tailored to the specific operation of LTE radios. The difficulty in designing such an algorithm is that the bundling decisions have to be made online without knowing the timing of future message transfer requests. For instance, many IoT applications generate messages in an event-based manner with no predictable schedule [29].

In Chapter 4, we formulated bundling as a cost minimization problem, where the cost function is a weighted summation of energy and delay costs. However, the energy cost was characterized using an On/Off radio model. As mentioned in our related work discussion on

DRX (subsection 2.2.3 in Chapter 2), the On/Off model does not accurately represent the effect of the extended DRX mechanism introduced in LTE to deal with IoT traffic. Thus, in this chapter, we incorporate the interplay between energy consumption and the extended DRX mechanism into our energy cost model. Also, in this chapter, we only focus on the max delay function to characterize the delay cost of the bundling algorithm. Accordingly, we adjust the implementation of the Break-Even (BE) algorithm proposed in Chapter 4 to account for the new energy cost model. We present a detailed analysis of BE in the new setting using the well-known notion of competitive ratio. We evaluate the performance of the proposed algorithm and the accuracy of our analysis in a range of realistic scenarios using both model-driven simulations and real experiments on an IoT testbed.

It is worth noting that while energy management in IoT applications was the main motivation for our work in this chapter, *the presented online algorithm and its analysis are applicable to any LTE device.*

The rest of the chapter is organized as follows. In Section 5.1 we present a formal specification of the problem. In Section 5.2 we describe an offline algorithm as a benchmark solution to the problem. We introduce our online algorithm in Section 5.3. Then we perform competitive analysis of the online algorithm in Section 5.4. Performance evaluation results are discussed in Section 5.5.

5.1 Problem Statement

The problem considered in this chapter is mostly similar to the one studied in previous chapter. The main difference is the adoption of a new function to characterize the energy cost, which reflects the effect of the DRX mechanism on energy consumption. Also, as mentioned before, in this chapter we only use max delay function to characterize the delay cost. We include a complete description of the problem here for the reader's convenience.

Consider a sequence of message transfer request arrivals $\mathcal{A} = \langle a_1, \ldots, a_n \rangle$, where a_i de-

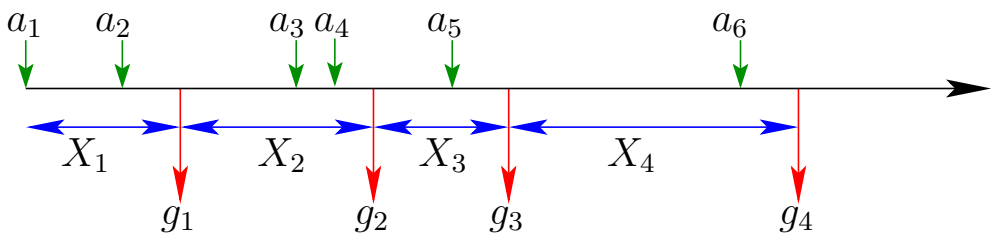


Figure 5.1: Relation between arrivals, grants and intervals.

notes the arrival time of request i. *The sequence $\mathcal{A}$ is not known in advance.* Without loss of generality, we assume that the radio is in idle state when the first request arrives. The goal is to design an online algorithm to bundle multiple requests together and grant them at once as opposed to individually granting each request. A message transfer request may involve uploading environmental readings received at an LTE-enabled IoT device from multiple IoT sensors. The LTE-enabled IoT device may communicate with its sensors using short-range low-power wireless technologies such as Bluetooth Low Energy (BLE) [29]. Let $\mathcal{G}_A = \langle g_1, \ldots, g_k \rangle$ denote the sequence of grants made by some algorithm A, for the arrival sequence $\mathcal{A}$, where g_i denotes the time of grant i. Let $\mathcal{X}_A = \{X_1, \ldots, X_k\}$ denote the set of all grant intervals of algorithm A, where $X_1 = [a_1, g_1]$ and $X_i = (g_{i-1}, g_i]$, for $i \geq 2$. All requests that arrive during the interval X_i are bundled together and granted at time g_i. Throughout the chapter, we use the notation X_i to refer to the i-th grant interval as well as the length of that interval, when there is no ambiguity.

Fig. 5.1 shows the relation between arrivals and grants. The objective of the bundling algorithm is to determine the grant times g_i that minimize the cost $C_A = E_A + \alpha D_A$, where E_A and D_A denote the *energy cost* and *delay cost* of algorithm A, respectively. The coefficient α is a control parameter that can be used to specify the relative importance of delay cost over energy cost depending on the IoT application requirements.

5.1.1 Energy Cost

The energy cost E_A is the tail energy consumed because of inactivity periods between grants of algorithm A. Let P_C and P_D denote the base powers consumed by the radio during the active and DRX substates of the RRC_Connected state, respectively, where $P_C > P_D$. Also, let T_i denote the length of the inactivity timer in active state and T_t denote the overall RRC tail time, where $T_t > T_i$. Similar to [106, 107], we use the following function to characterize the tail energy:

$$\varepsilon(\tau) = \begin{cases} P_C\tau & 0 \leq \tau \leq T_i, \\ P_C T_i + P_D(\tau - T_i) & T_i < \tau \leq T_t, \\ P_C T_i + P_D(T_t - T_i) & T_t < \tau, \end{cases} \tag{5.1}$$

where, τ is the time passed since the last grant of the algorithm. Then, the energy cost of grant interval X_i is given by $E_A(X_i) = \varepsilon(X_i)$. Consequently, the energy cost of the algorithm A is given by,

$$E_A = \sum_{X_i \in \mathcal{X}_A} E_A(X_i) + \varepsilon(T_t), \tag{5.2}$$

where the additional term $\varepsilon(T_t)$ is added to account for a tail time after the last grant of the algorithm. To simplify the analysis, similar to [16, 76], we have ignored the transfer time of bundles as this time is the same for every bundling algorithm.

5.1.2 Delay Cost

The delay cost of the algorithm is defined as the sum of delay costs of all the bundles. We use the notation $D_A(X_i)$ to denote the delay cost of bundle i, which includes all requests that arrive during interval X_i. Consider a request $a_j \in X_i$. The delay cost of request a_j is given by $(g_i - a_j)$. The delay cost of bundle i is then expressed as $D_A(X_i) = \max_{a_j \in X_i}(g_i - a_j)$. In other words, the delay cost of a bundle is the maximum of all the delays of the requests in the bundle. Equivalently, the delay cost of bundle i is given by $g_i - a_{first,i}$, where $a_{first,i}$

is the arrival time of the first request in bundle i. It then follows that,

$$D_A = \sum_{X_i \in \mathcal{X}_A} D_A(X_i) = \sum_{X_i \in \mathcal{X}_A} \max_{a_j \in X_i} (g_i - a_j)\,. \tag{5.3}$$

5.2 Optimal Offline Algorithm

As a point of comparison for our online algorithm, we design an optimal offline algorithm, called OPT. OPT is not a realistic algorithm, since *it knows the entire request arrival sequence in advance*. An important observation used in designing OPT is the fact that the optimal algorithm always makes grants right at the time of some request arrivals and never makes a grant in-between two arrivals. Based on this observation, we design OPT using a dynamic programming algorithm similar to the one in [76]. OPT has the runtime of $O(n^2)$, where n is the length of the arrival sequence. A detailed discussion of the optimal offline algorithm and its analysis can be found in Section 4.2 of Chapter 4.

Theorem 4. *When $\alpha \geq P_C$,* OPT *makes a grant for every request arrival.*

Proof. The proof of this Theorem is exactly the same as the proof of Theorem 1 in Chapter 4. The only difference is that the comparison between the increase in energy cost and the decrease in weighted latency cost, now relies on the fact that when $\alpha > P_C$, we have $\alpha\tau > \varepsilon(\tau)$.

Also, similar to the proof of Theorem 1, it can be established that when $\alpha = P_C$, there exist multiple optimal grant sequences and that granting at each arrival is one of them. ■

5.3 Online Energy Management Algorithm

With Theorem 4 characterizing the behavior of the optimal algorithm for $\alpha \geq P_C$, our online algorithm called Break-Even (BE) will imitate OPT in that regime. In other regimes, BE works as follows. Assume that the most recent grant was at time g_i and the algorithm has to decide when to make its next grant g_{i+1}. Let τ denote the time duration since the last grant

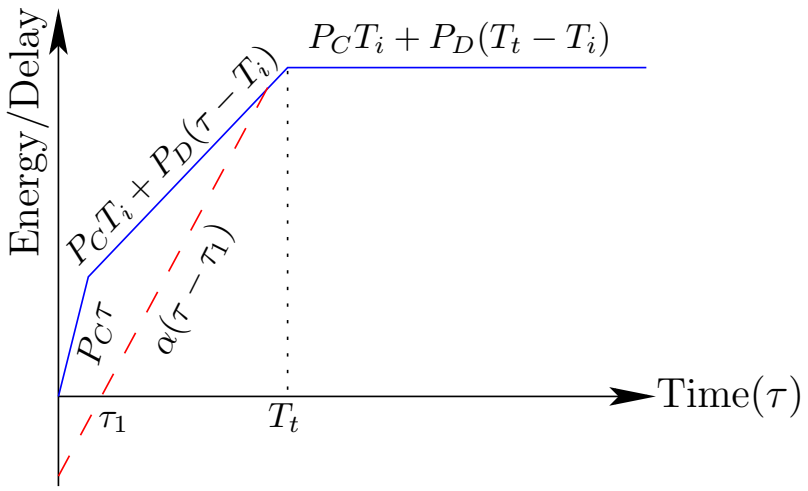


Figure 5.2: Intersection of energy and delay functions.

g_i. Also, let $E_{\text{BE}}(\tau)$ and $D_{\text{BE}}(\tau)$ denote the energy and delay cost incurred by BE during τ. Then, BE makes a grant at time $g_{i+1} = g_i + \tau$, when $E_{\text{BE}}(\tau) = \alpha D_{\text{BE}}(\tau)$ holds. Fig. 5.2 portrays a plot illustrating the behavior of BE. In this figure, τ_1 is the time between the last grant of the algorithm (*i.e.*, g_i) and the first arrival after that. This way, the weighted delay cost associated with the grant at $g_i + \tau$ will be $\alpha(\tau - \tau_1)$, which is presented by the dashed line (called D-line). The solid polyline (called E-line) is the graphical presentation of the energy cost function defined in (5.1). The intersection of these two lines determines the desired τ value.

In the following sections, we will focus on the value of $\tau - \tau_1$ to analyse the performance of BE. If $\alpha \leq P_D$, D-line will intersect horizontal part of the E-line, and hence,

$$\tau - \tau_1 = \frac{P_C T_i + P_D(T_t - T_i)}{\alpha} = \frac{\varepsilon(T_t)}{\alpha}, \qquad \text{if } \alpha \leq P_D\,. \tag{5.4}$$

If $P_D < \alpha < P_C$, D-line can intersect the middle part or the horizontal part of the E-line. Specifically, we have,

$$\tau - \tau_1 = \begin{cases} f(\tau_1), & \text{if } (P_C - P_D)T_i + \alpha\tau_1 \leq (\alpha - P_D)T_t, \\ \frac{\varepsilon(T_t)}{\alpha}, & \text{otherwise,} \end{cases} \tag{5.5}$$

where, $f(\tau_1) = \frac{(P_C - P_D)T_i + P_D\tau_1}{\alpha - P_D}$. Notice that the first grant is treated differently, *i.e.*, when

there is no g_i. The algorithm makes its first grant at some time τ that satisfies the equation $\varepsilon(T_t) = \alpha D_{\mathrm{BE}}(\tau)$.

The algorithm BE can be implemented using timers. Specifically, for the first arrival after each grant g_i, BE sets a timer to time out after w time units, where the value of w can be computed from (5.4) or (5.5) depending on the values of α, τ_1, and their relations to the power model parameters. Upon expiry of the timer, a grant will be made and all the pending requests will be granted.

5.4 Analysis of the Break-Even Algorithm

As mentioned earlier, we will study the behaviour of BE in three different regimes. In the regime of $\alpha \geq P_C$, BE imitates the behavior of OPT as determined by Theorem 4. As such, BE is 1-competitive when $\alpha \geq P_C$. Therefore, it suffices to analyze BE for the remaining two regimes, namely when $\alpha \leq P_D$ and $P_D < \alpha < P_C$. In the sequel, we focus on proving the following theorems.

Theorem 5. *When $\alpha \leq P_D$, the Break-Even algorithm is 2-competitive.*

Theorem 6. *When $P_D < \alpha < P_C$, the Break-Even algorithm is 4-competitive.*

5.4.1 Preliminaries

The following observations will be used in our analysis.

Observation 3. *At least one request arrives during an interval $X_i = (g_{i-1}, g_i]$. If there is no arrival, then the algorithm does not make any grant at g_i as there is no request to grant.*

Observation 4. *Energy function $\varepsilon(\tau)$ is a concave piecewise-linear function [108], where*

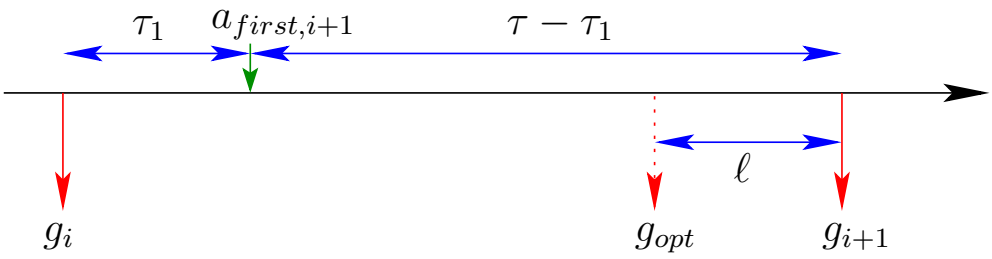


Figure 5.3: $(g_i, g_{i+1}]$ is a sample interval created by BE.

the following relations always hold,

$$\varepsilon(\tau) \leq P_C \tau, \tag{5.6}$$

$$\varepsilon(\tau) \leq P_C T_i + P_D(\tau - T_i), \tag{5.7}$$

$$\varepsilon(\tau) \leq P_C T_i + P_D(T_t - T_i) = \varepsilon(T_t)\,. \tag{5.8}$$

5.4.2 Analysis of a Single Interval

We focus on individual grant intervals created by BE in isolation. Fig. 5.3 shows one such interval, which we refer to as X. Considering the distance of $(\tau - \tau_1)$ between the first arrival in X and its associated grant, the weighted delay cost incurred by BE will be $\alpha(\tau - \tau_1)$. Since BE makes a grant when $E_{\text{BE}}(X) = \alpha D_{\text{BE}}(X)$, the cost of interval X is,

$$C_{\text{BE}}(X) = 2\alpha D_{\text{BE}}(X) = 2\alpha(\tau - \tau_1) = 2E_{\text{BE}}(X)\,. \tag{5.9}$$

If OPT does not make any grant in X, it will incur at least a delay cost equal to $D_{\text{BE}}(X)$, and hence $C_{\text{OPT}}(X) \geq \alpha D_{\text{BE}}(Y)$, which establishes $C_{\text{BE}}(X) \leq 2C_{\text{OPT}}(X)$. In case OPT has at least one grant in X, we will use g_{OPT} to refer to its first grant in this interval. Let ℓ denote the time duration from g_{OPT} up to the end of interval X (Fig. 5.3). We will charge g_{OPT} with $\varepsilon(\ell)$ for its contribution to the energy cost and $\alpha(\tau - \tau_1 - \ell)$ for its contribution to the weighted delay cost. If $\ell > T_t$, then $\varepsilon(\ell) = \varepsilon(T_t) \geq E_{\text{BE}}(X)$, indicating $2C_{\text{OPT}}(X) \geq C_{\text{BE}}(X)$. Otherwise (*i.e.*, if $\ell \leq T_t$), we have,

$$C_{\text{OPT}}(X) \geq \alpha(\tau - \tau_1 - \ell) + \varepsilon(\ell)$$

$$= \begin{cases} \alpha(\tau - \tau_1) + (P_C - \alpha)\ell & \text{if } 0 \leq \ell \leq T_i, \\ \alpha(\tau - \tau_1) + (P_C - P_D)T_i & \\ \quad + (P_D - \alpha)\ell & \text{if } T_i < \ell \leq \tau - \tau_1\,. \end{cases} \tag{5.10}$$

$C_{\text{OPT}}(X)$ is lower bounded in (5.10) because it is possible for OPT to have more than one grant in interval X.

Proof of Theorem 5:

When $\alpha \leq P_D$ in (5.10), ℓ's coefficient in both cases becomes positive (also $P_C - P_D > 0$). As a result, it always holds that $\alpha(\tau - \tau_1) \leq C_{\text{OPT}}(X)$, which implies that $C_{\text{BE}}(X) \leq 2C_{\text{OPT}}(X)$.

■

Notice that when $P_D < \alpha < P_C$, the coefficient of ℓ in (5.10) is positive only in the first case. As a result, we have $\alpha(\tau - \tau_1) \leq C_{\text{OPT}}(X)$ only in the first case, and hence $C_{\text{BE}}(X) \leq 2C_{\text{OPT}}(X)$. On the other hand, ℓ's coefficient is negative in the second case of (5.10). Therefore, the minimum value of the lower bound in (5.10) is obtained by the maximum possible value for ℓ, which is given by $\min\{\tau - \tau_1, T_t\}$. If $T_t < \tau - \tau_1$, we obtain that

$$C_{\text{OPT}}(X) \geq \alpha(\tau - \tau_1 - T_t) + \varepsilon(T_t) \geq E_{\text{BE}}(X), \tag{5.11}$$

which results in $C_{\text{BE}}(X) \leq 2C_{\text{OPT}}(X)$. However, if $T_t \geq \tau - \tau_1$, we obtain that,

$$C_{\text{OPT}}(X) \geq P_D(\tau - \tau_1) + (P_C - P_D)T_i\,. \tag{5.12}$$

5.4.3 Cost of Grant Intervals

In the previous subsection, we studied BE intervals in isolation. Specifically, in each BE interval we charged BE with the energy cost of the entire interval. In contrast, we charged potential OPT grants only with the energy cost of a portion of the interval (*i.e.*, $\tau - \tau_1$),

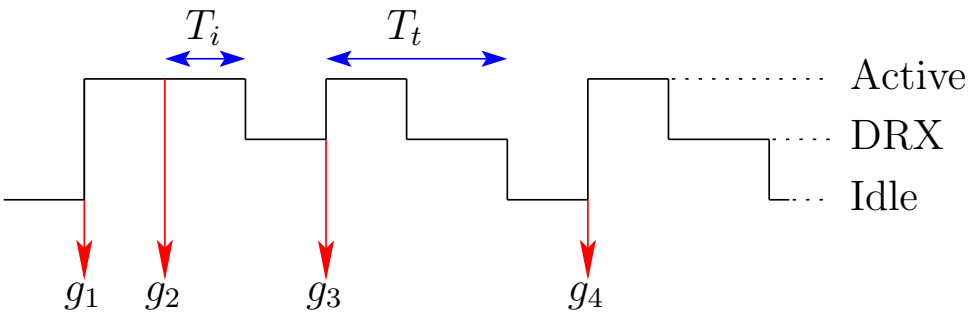


Figure 5.4: Grants and radio state transitions of OPT.

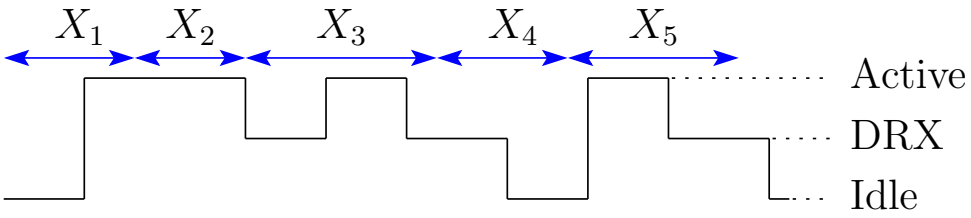


Figure 5.5: Grant intervals of BE overlaid on radio states of OPT.

ignoring energy consumption during τ_1 period. This can result in a pessimistic bound for the competitive ratio. Thus, in this section, we study the entire set of intervals altogether.

We consider $\mathcal{G}_{OPT} = \langle g_1, \ldots, g_l \rangle$ to be the grants made by OPT. Because of these grants, the radio will transition between different states (idle, DRX and active state) several times. Fig. 5.4 depicts an example consisting of a few OPT grants and radio state transitions resulting from them. In general, the radio is initially in the idle state, then goes through several transitions and finally goes to the idle state after T_t time from the last grant. The main technique used in this section is to overlay grant intervals of BE over the radio states under OPT (see Fig. 5.5). This way, we can identify two classes of intervals:

1. Intervals with no radio state transition,

2. Intervals with one or more radio state transitions.

The analysis of these interval classes is presented in the next subsections.

5.4.4 Intervals with No Radio Transition

Let X represent one such interval. Considering the radio state during X, the following cases can be identified.

1. *Radio is in active state during interval X:* In this case, we only focus on the energy cost incurred by OPT as its delay cost could be as low as zero. Since the OPT radio is in the active state during interval X, we have $E_{\mathrm{OPT}}(X) = P_C X$. Based on Observation 4, it is obtained that,

$$C_{\mathrm{OPT}}(X) \geq P_C X \geq E_{\mathrm{BE}}(X)\,. \tag{5.13}$$

   Recall that $C_{\mathrm{BE}}(X) = 2E_{\mathrm{BE}}(X)$, which yields $2C_{\mathrm{OPT}}(X) \geq C_{\mathrm{BE}}(X)$.

2. *Radio is in DRX or idle states during interval X:* Based on Observation 3, the fact that BE makes a grant at the end of interval X implies the arrival of at least one request during X. When the radio for OPT spends the entire interval X in the DRX or idle states, it means that OPT did not make a grant during X, because otherwise it would have transitioned to the active state. Therefore, OPT incurs at least a delay cost equal to $D_{\mathrm{BE}}(X)$. Thus, we have,

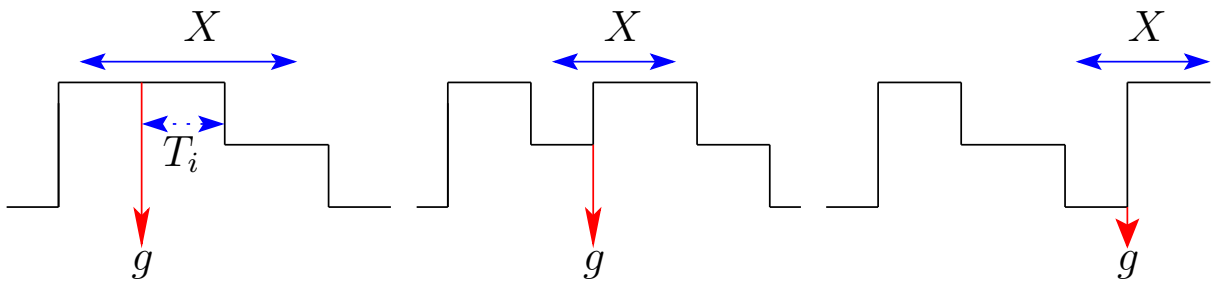
$$C_{\mathrm{OPT}}(X) \geq \alpha D_{\mathrm{BE}}(X)\,. \tag{5.14}$$

   Combining (5.14) with $C_{\mathrm{BE}}(X) = 2\alpha D_{\mathrm{BE}}(X)$ results in $2C_{\mathrm{OPT}}(X) \geq C_{\mathrm{BE}}(X)$.

5.4.5 Intervals with One or More Radio Transitions

Let X refer to one such interval. If OPT does not have a grant in X then $C_{\mathrm{BE}}(X) \leq 2C_{\mathrm{OPT}}(X)$ will hold because OPT will suffer from at least the delay cost $D_{\mathrm{BE}}(X)$. Therefore, in the following, we assume OPT makes at least one grant during interval X.

When the first grant of OPT in X happens, the OPT radio can be in any of the following three possible power states.



(a) First grant of OPT happens when the radio is in active state.

(b) First grant of OPT happens when the radio is in DRX state.

(c) First grant of OPT happens when the radio is in idle state.

Figure 5.6: Intervals with one or more radio transitions.

1. *The first grant happens when the radio is in active state:* Fig. 5.6(a) depicts this case. Since OPT makes grants only at request arrival times (and not in-between them), we know that OPT can not have a grant during the period τ_1. Also notice that only a grant can cause the radio to transition to the active state (in contrast to the expiry of an RRC timer and state demotion). As such, the radio of OPT should be in the active state from the beginning of X until the first OPT grant in X (and this time period covers τ_1).

   As shown in subsection 5.4.2, $C_{\text{OPT}}(X)$ and $C_{\text{BE}}(X)$ always satisfy the relation $2C_{\text{OPT}}(X) \geq C_{\text{BE}}(X)$, except in one case. The only case where this relation does not hold leads to the lower bound in (5.12). Given that this lower bound is obtained without accounting for the radio energy consumption during τ_1, we can adjust the lower bound by considering the fact that the OPT radio will be in the active state during the period τ_1. Therefore, given that $P_C > P_D$, we have,

$$\begin{aligned} C_{\text{OPT}}(X) &\geq P_D(\tau - \tau_1) + (P_C - P_D)T_i + P_D\tau_1, \\ &= P_C T_i + P_D(\tau - T_i), \\ &\geq E_{\text{BE}}(X)\,. \end{aligned} \tag{5.15}$$

   The last inequality comes form (5.7) in Observation 4 by considering that τ represents the length of interval X. Finally, the relation $C_{\text{BE}}(X) = 2E_{\text{BE}}(X)$ yields $2C_{\text{OPT}}(X) \geq C_{\text{BE}}(X)$.

2. *The first grant happens when the radio is in the DRX state:* Fig. 5.6(b) depicts this case. This case is similar to the previous case. Notice that the only way the radio can transition

to the DRX state is through the expiry of the RRC inactivity timer and demotion from the active state. In other words, making a grant will always bring the radio to the active state and not the DRX state. Thus, the only difference compared to the previous case is that the OPT radio can be in the active and DRX states during τ_1 but not in the idle state. As a result, the lower bound in (5.12) can be adjusted by adding $P_D\tau_1$, which means that the relation in (5.15) still holds. Therefore, we have $2C_{\text{OPT}}(X) \geq C_{\text{BE}}(X)$.

3. *The first grant happens when the radio is in the idle state:* Fig. 5.6(c) depicts this case. In this case, OPT's first grant (called g_{OPT}) could be right at the arrival time of the first request in the interval, which implies zero delay cost. On the other hand, if g_{OPT} is close to the end of interval X, the energy cost incurred by OPT during interval X could also be zero. As a result, the cost of $C_{\text{OPT}}(X)$ could be as low as zero. Based on Observation 4, the following upper bound is obtained for $C_{\text{BE}}(X)$,

$$C_{\text{BE}}(X) = 2E_{\text{BE}}(X) \leq 2\varepsilon(T_t)\,. \tag{5.16}$$

Let $\mathcal{X}^0_{\text{BE}} \subseteq \mathcal{X}_{\text{BE}}$ denote the set of all such intervals of BE. Instead of establishing a lower bound for C_{OPT} over individual intervals, we will bound the cost of OPT over the entire set of such intervals (*i.e.*, over $\mathcal{X}^0_{\text{BE}}$). This will be discussed in the proof of Theorem 6 presented next.

Proof of Theorem 6:

For computing C_{OPT} and C_{BE}, we will use the following:

$$C_{\text{OPT}} = \sum_{X_i \in \mathcal{X}_{\text{BE}}} C_{\text{OPT}}(X_i) + \varepsilon(T_t), \tag{5.17}$$

$$C_{\text{BE}} = \sum_{X_i \in \mathcal{X}_{\text{BE}}} C_{\text{BE}}(X_i) + \varepsilon(T_t)\,. \tag{5.18}$$

Based on the analysis in previous subsections, every interval X_i in $\mathcal{X}_{\text{BE}}/\mathcal{X}^0_{\text{BE}}$ satisfies $C_{\text{BE}}(X_i) \leq 2C_{\text{OPT}}(X_i)$. Also based on (5.16), every interval X_i in $\mathcal{X}^0_{\text{BE}}$ satisfies $C_{\text{BE}}(X_i) \leq 2\varepsilon(T_t)$.

Therefore, we have,

$$\begin{aligned} C_{\mathrm{BE}} &= \sum_{X_i \in \mathcal{X}_{\mathrm{BE}}} C_{\mathrm{BE}}(X_i) + \varepsilon(T_t) \\ &= \sum_{X_i \in \mathcal{X}_{\mathrm{BE}} \setminus \mathcal{X}^0_{\mathrm{BE}}} C_{\mathrm{BE}}(X_i) + \sum_{X_i \in \mathcal{X}^0_{\mathrm{BE}}} C_{\mathrm{BE}}(X_i) + \varepsilon(T_t) \\ &\leq 2 \sum_{X_i \in \mathcal{X}_{\mathrm{BE}} \setminus \mathcal{X}^0_{\mathrm{BE}}} C_{\mathrm{OPT}}(X_i) + \sum_{X_i \in \mathcal{X}^0_{\mathrm{BE}}} 2\varepsilon(T_t) + \varepsilon(T_t) \\ &\leq 2C_{\mathrm{OPT}} + 2\varepsilon(T_t)\,|\mathcal{X}^0_{\mathrm{BE}}|, \end{aligned} \tag{5.19}$$

where, $|\mathcal{X}^0_{\mathrm{BE}}|$ denotes the cardinality of set $\mathcal{X}^0_{\mathrm{BE}}$. Thus, our analysis is reduced to establishing an upper bound on $|\mathcal{X}^0_{\mathrm{BE}}|$. To this end, we focus on the behavior of OPT and observe that for the entire arrival sequence, the OPT radio will transition between different states several times. Assume that for $\mathcal{K}$ times, there is a transition from idle to active state. Accordingly, we have $C_{\mathrm{OPT}} \geq \mathcal{K}\varepsilon(T_t)$, because every time the radio goes to the active state, it incurs at least the energy cost of $\varepsilon(T_t)$ before going back to the idle state. Also notice that after overlaying BE intervals over the radio states under OPT, we cannot have more than $\mathcal{K}$ intervals belonging to $\mathcal{X}^0_{\mathrm{BE}}$. Thus, it is obtained that,

$$|\mathcal{X}^0_{\mathrm{BE}}| \leq \mathcal{K} \leq \frac{C_{\mathrm{OPT}}}{\varepsilon(T_t)}\,. \tag{5.20}$$

By replacing $|\mathcal{X}^0_{\mathrm{BE}}|$ in (5.19) with its upper bound from (5.20), the following relation is obtained,

$$\begin{aligned} C_{\mathrm{BE}} &\leq 2C_{\mathrm{OPT}} + 2\varepsilon(T_t)|\mathcal{X}^0_{\mathrm{BE}}|, \\ &\leq 2C_{\mathrm{OPT}} + 2C_{\mathrm{OPT}} = 4C_{\mathrm{OPT}}\,. \end{aligned} \tag{5.21}$$

■

5.4.6 Remarks on the competitive ratio of Theorem 6

We note that the competitive ratio of 4 proved in Theorem 6 is not tight. This can be observed by the discussions in subsection 5.4.2, where grant intervals were considered in

isolation. Specifically, using (5.9) and (5.12), the following upper bound for $\frac{C_{\text{BE}}}{C_{\text{OPT}}}$ can be established,

$$\begin{aligned} \frac{C_{\text{BE}}}{C_{\text{OPT}}} &\leq \frac{2\alpha(\tau - \tau_1)}{P_D(\tau - \tau_1) + (P_C - P_D)T_i} \\ &\leq \frac{2\alpha(\tau - \tau_1)}{P_D(\tau - \tau_1)} = \frac{2\alpha}{P_D}. \end{aligned} \tag{5.22}$$

This implies that, for example, when α/P_D is 1.5, the competitive ratio of BE is bounded by 3.

5.5 Performance Evaluation

In this section, we evaluate BE using both model-driven simulations and real experiments on an IoT testbed. We compare BE with two algorithms: 1) OPT, and 2) *Default*, which grants requests as soon as they arrive. Notice that the delay cost of the Default is always zero.

5.5.1 Model-Driven Evaluation

In this part, we study the performance of different algorithms using a custom-developed discrete-event simulator. The simulator takes as input the weight factor α, parameters of the power model (T_i, T_t, P_C, P_D) and the transfer request arrival sequence. Unless otherwise stated, parameters of the power model are set to the values reported in Table 5.1. These values are reported in [14] based on measurements in an LTE network. The DRX base power (P_D) is computed by taking the weighted average of LTE tail base power and power consumption of ON durations in each DRX cycle.

Table 5.1: Power model parameters.

State	Power (mW)	Duration (ms)
Active	$P_C = 788$	$T_i = 200$
DRX	$P_D = 163$	$T_t = 11000$

5.5.1.1 Exploring Energy-Delay Tradeoff

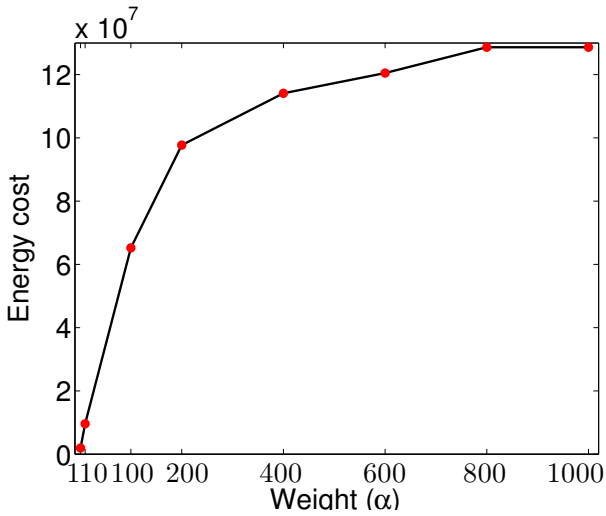
We used a sequence of size 100 requests with normal inter-arrival times (μ = 7000 ms, σ = 6000 ms) to perform this experiment. About 2% and 68% of the inter-arrival times in the sequence are less than T_i and T_t, respectively.

Figs. 5.7(a) and 5.7(b) show the energy and delay costs for different values of α, respectively. These two plots are combined in Fig. 5.7(c) which shows the pairwise energy and delay values next to their corresponding weight factors. It is observed that the energy consumed by BE decreases with lower values of the weight α. For example, $\alpha = 1$ results in 98.5% energy saving compared to $\alpha = 800$. Fig. 5.7(d) which shows the average size of the bundles, illustrates BE's ability to adapt its behavior depending on the weight assigned to the delay cost. Fig. 5.7(e) presents the cumulative distribution function (CDF) of the delay experienced by the individual requests in the sequence. Notice that in our considered cost function, the delay cost (D_{BE}) is defined as the summation of the maximum delays experienced in each bundle. However, as we can see in Fig. 5.7(e), D_{BE} is directly related to the delays experienced by individual requests.

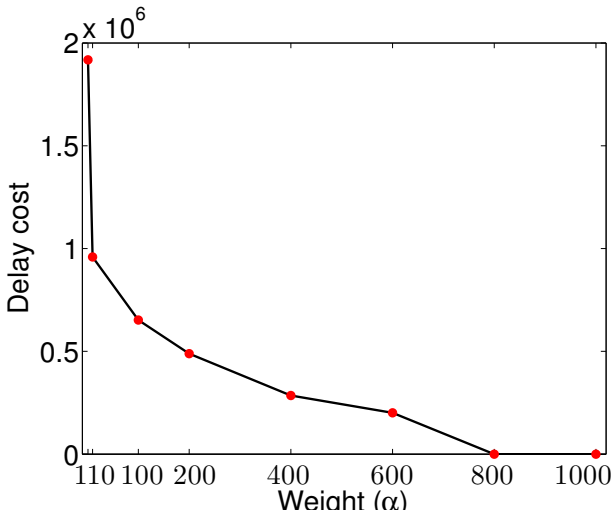
The empirical competitive ratios for different values of α are listed in Table 5.2. These results conform the properties claimed in Theorems 4, 5, and 6. Also in the settings characterized by $P_D < \alpha < P_C$, BE exhibits a performance significantly better than the one predicted by the competitive ratio of 4.

Table 5.2: Empirical competitive ratio of BE.

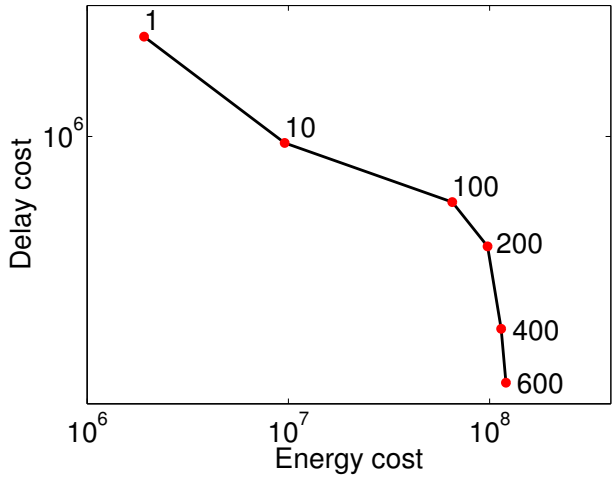
α	$C_{\text{BE}}/C_{\text{OPT}}$	α	$C_{\text{BE}}/C_{\text{OPT}}$
1	1.41	400	1.78
10	1.93	600	1.87
100	1.59	800	1
200	1.52	1000	1



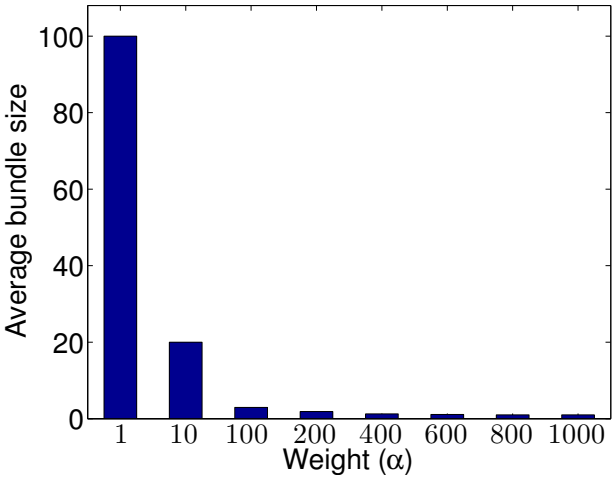
(a) Energy cost (in micro-Joules).



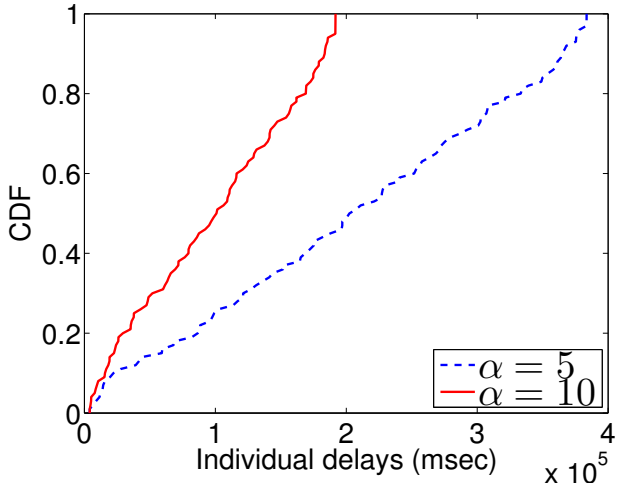
(b) Delay cost (in ms).



(c) Energy-delay tradeoff.



(d) Average bundle size.



(e) CDF of individual delays.

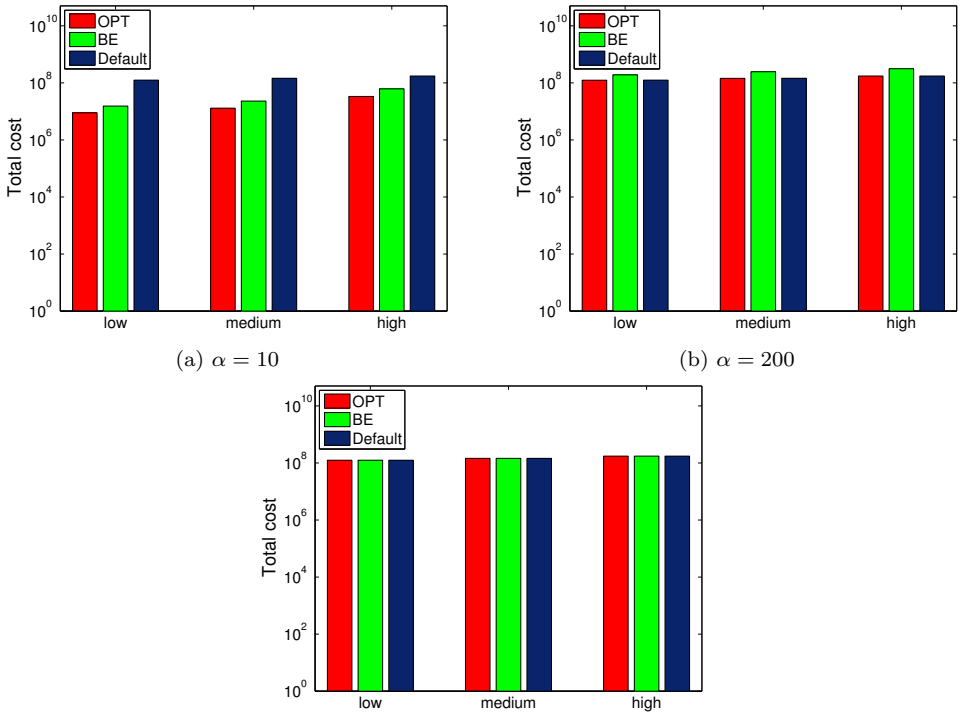
Figure 5.7: Performance of BE: By controlling α, different energy-delay tradeoffs can be achieved.

5.5.1.2 Performance under Different Arrival Patterns

Similar to [101], here we change the fluctuation level of the inter-arrival times to generate different patterns of request arrivals. In particular, based on the coefficient of variation (CV) of inter-arrival times, we consider arrival sequences of *low* (CV = 0.5), *medium* (CV = 1.5) and *high* (CV = 5) fluctuations. The inter-arrival times are normally distributed with mean 7000 ms.

Fig. 5.8 shows the total cost achieved with the three algorithms under varying fluctuation levels. We consider three weight values corresponding to three regimes identified by Theorems 4, 5, and 6. In Fig. 5.8(b), the cost of BE is 1.53 and 1.81 times the cost of OPT for sequences with low and high fluctuation, respectively. This implies that for a specific delay weight, the performance of BE changes depending on the characteristics of the arrival sequence. The total costs presented in Fig. 5.8 also verify our analysis, since the maximum value of $C_{\text{BE}}/C_{\text{OPT}}$ is 1.84 among all the considered scenarios. In scenarios where energy is more important ($\alpha \leq P_D$), BE outperforms the Default algorithm. For example, in a setting with $\alpha = 10$ and high fluctuation, BE results in 64.6% reduction in the total cost compared to Default.

Across all α values, BE's worst performance is achieved when $P_D < \alpha < P_C$. While in this regime BE results in lower energy consumption compared to the Default algorithm, the higher weight assigned to the delay causes the total cost of BE to be higher. In this regime, Default performs better in sequences with long inter-arrival times, where most of the gaps are longer than BE's timer value. In that case, not only BE misses chances of bundling, but also incurs higher cost due to unnecessary waiting. In contrast, BE outperforms Default in sequences with shorter inter-arrival times. For example, in an experiment characterized by $\alpha = 200$ and normal inter-arrival times of mean 400 ms and standard deviation 200 ms, the Default's total cost is 14% higher than BE's cost. As Fig. 5.8(c) illustrates, in scenarios with high delay importance ($\alpha \geq P_C$), all three algorithms have an identical performance as they



(a) $\alpha = 10$

(b) $\alpha = 200$

(c) $\alpha = 800$

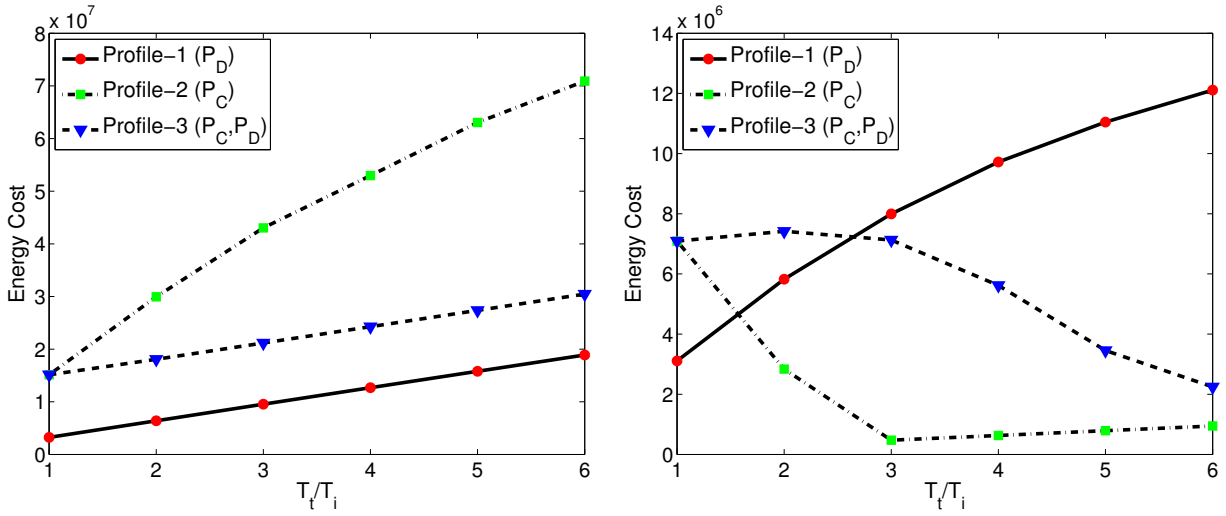
Figure 5.8: Comparing the performance of BE with OPT and Default under different fluctuation levels of request inter-arrival times.

grant requests as soon as they arrive.

5.5.1.3 Comparison with On/Off Radio Models

To study the difference between the LTE radio model and an On/Off model, we compare the performance of BE under 3 different power profiles. Specifically, **Profile-1** and **Profile-2** are On/Off radio models, where the radio consumes P_D and P_C for the entire tail period (T_t), respectively. **Profile-3** represents the LTE radio model characterized by parameters P_C, P_D, T_i, T_t.

We performed experiments using a sequence of size 100 with normal inter-arrival times ($\mu = 7000$ ms, $\sigma = 6000$ ms). For P_C, P_D and T_i we used values reported in Table 5.1, but we changed T_t between 200 and 1200 ms. For $\alpha = 200$, Fig. 5.9(a) shows the energy cost of BE under the three power profiles as a function of tail time ratio T_t/T_i (called **TTR**). As seen, for all the TTR values, **Profile-2** and **Profile-1** result in the highest and the lowest



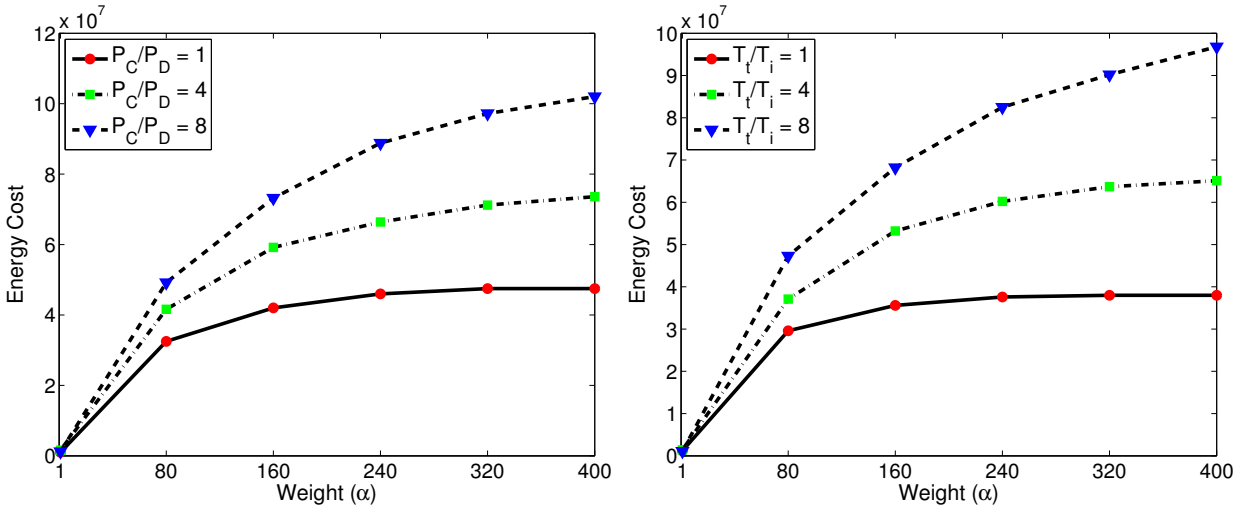
(a) BE's energy cost under long inter-arrival times

(b) OPT's energy cost under short inter-arrival times

Figure 5.9: Energy cost under different power models.

energy consumption, respectively.

We also observed a behavior similar to the one in Fig. 5.9(a) when using input sequences with short inter-arrival times. However, depending on the characteristics of the input sequence, OPT can exhibit a different behaviour. For example, Fig. 5.9(b) compares the performance of OPT under the three power profiles using an input sequence with short inter-arrival times (normal with $\mu = 700$ ms and $\sigma = 600$ ms). We can see a different ordering between power profiles as **Profile-2** results in the lowest energy consumption. This stems from the fact that OPT can make bundling decisions based on its knowledge about the entire sequence. In particular, when arrival times are close to each other, the delay cost of bundling would be low compared to the reduction in its energy cost. Thus, with an increase in power dissipation rate (**Profile-2**), OPT more aggressively reduces the number of grants resulting in low inter-grant time gaps. In contrast, when inter-arrival times are longer, OPT cannot adopt such an aggressive policy as it would result in high delay costs. That is why OPT's energy cost follows a similar trend to the one in Fig. 5.9(a) under sequences with longer inter-arrival times.



(a) Energy cost for different ratios of $\frac{P_C}{P_D}$. $(P_D, T_i, T_t) = (500, 200, 1000)$.

(b) Energy cost for different ratios of $\frac{T_t}{T_i}$. $(P_D, P_C, T_i) = (500, 2000, 200)$.

Figure 5.10: Energy cost under different ratios of parameters.

5.5.1.4 Effect of Power Model Parameters

In this experiment, we study the effect of power model parameters (P_C and P_D) on the performance of BE. Specifically, we examine the performance of BE under different ratios of P_C/P_D (called power ratio) by using a fixed value for P_D and changing values of P_C. To better capture the effect of power ratio on BE's energy cost, we use the power model parameters characterized by $P_D = 500$ mW, $T_i = 200$ ms, and $T_t = 1000$ ms, which are different than the ones in Table 5.1. We performed experiments using a sequence of 100 requests with normal inter-arrival times ($\mu = 7000$ ms, $\sigma = 6000$ ms). Given the high importance of energy in IoT scenarios, we consider the regime of $\alpha \leq P_D$, where energy is more important than delay.

Fig. 5.10(a) plots the energy cost of BE under different power ratios as a function of the weight factor α. We can see that increasing the power ratio leads to higher energy consumption. Also notice that the increase in the energy cost becomes more pronounced in settings with higher values of α. This is because with an increase in delay importance, BE tends to avoid bundling and grants requests as soon as they arrive. This will create more inter-grant idle gaps which in turn will highlight the role played by a higher power

dissipation rate.

5.5.1.5 Effect of Tail Times

Here we study the effect of timers T_i and T_t on the performance of BE. Specifically, we perform experiments under three different ratios of **TTR** by using a fixed value for T_i and changing values of T_t. As in the previous subsection, we use parameter values of $(P_D, P_C, T_i) =$ (500 mW, 2000 mW, 200 ms) to better represent the impact of timers on the performance of BE. Also, we perform these experiments in the regime of $\alpha \leq P_D$ using the same sequence described in the previous subsection.

Fig. 5.10(b) depicts the energy cost for different values of TTR as a function of the delay weight (α). As observed, BE's energy cost is influenced by the values of the timers. Specifically, increasing TTR contributes to a higher energy consumption in all cases (α values). For example, in the case of $\alpha = 80$, raising TTR from 1 to 8 leads to 59.8% increase in the energy cost. This is due to the fact that with larger values of T_t, the radio stays longer in the DRX state instead of switching to the idle state. Similar to the previous section, in Fig. 5.10(b), an increase in the delay weight intensifies the impact of long tail time, which is the result of higher number of grants and longer inter-grant idle gaps.

5.5.2 Experiments on IoT Testbed

To assess the performance of BE in real-life conditions, we also performed experiments on Grenoble platform of the FIT IoT-LAB testbed [109]. IoT-LAB is a large scale open testbed for IoT research which provides access to IoT devices with IEEE 802.15.4-based radio transmitters. We created a topology consisting of 30 M3 Open nodes configured with Contiki operating system. Fig. 5.11 shows the topology used for this experiment. One of the nodes (node 231 in Grenoble platform) was configured to act as a gateway and the rest of the nodes were configured with a program that periodically (every 60 seconds) reads the value of atmospheric pressure from node's sensor and sends it to the gateway over a UDP

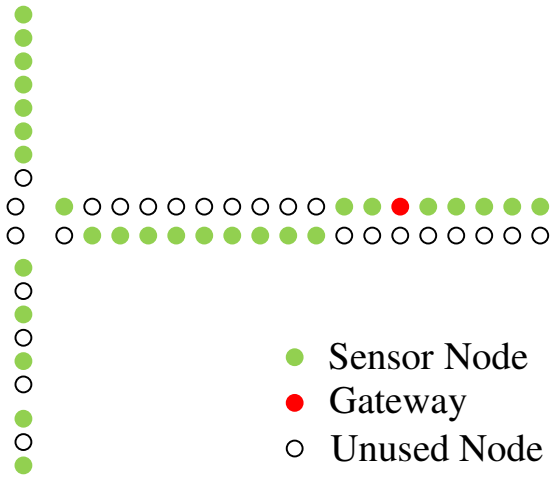
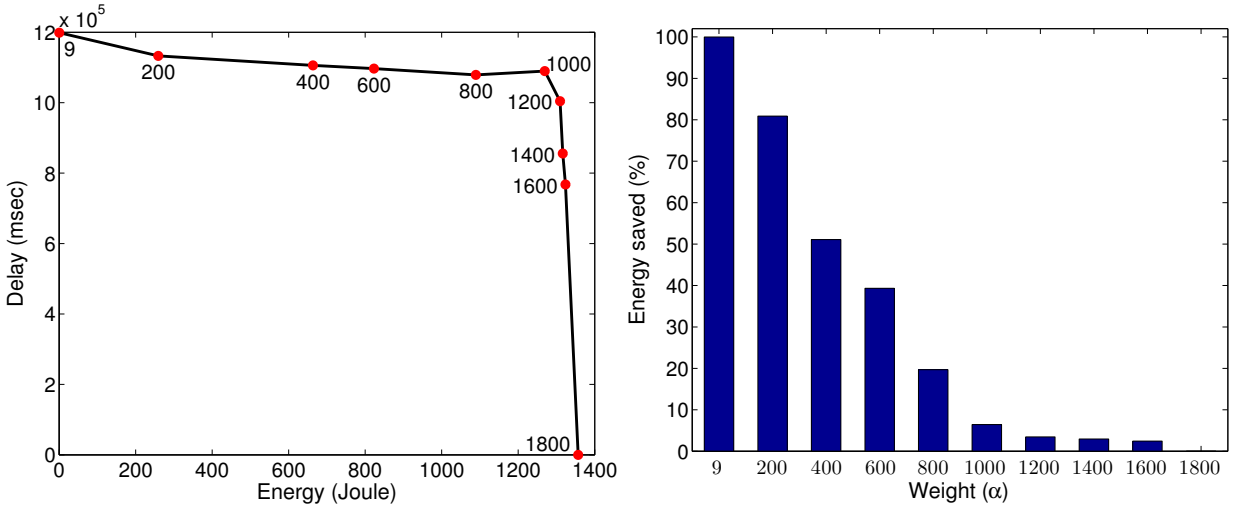


Figure 5.11: Topology of the experiment run on IoT-LAB testbed.

connection. As mentioned in [110], upon experiment initialization, each node goes through a slightly different setup phase required for establishing the routing-tree structures in the network. This creates a random delay before each node starts generating traffic which is one of the reasons for desynchronization among nodes. For a duration of 20 minutes, we captured radio communications at the gateway and created a trace from packet arrival times at the gateway.

Then for different values of α, we run BE and Default algorithms with the collected trace as their input sequence. For each value of α, we recorded the resulting grant times in a separate file. We then fed those grant files to our Android app installed on a Nexus smartphone. This app, which is developed for the purpose of radio energy measurement, performs message transfers at user-specified times. Each *run* of the app uses a grant sequence file as input. It then repeatedly sends message transfer requests at the times specified in the grant file. We also blocked all background traffic from OS services and other apps. To measure the energy consumption of the radio interface, we used the AT&T ARO tool [11], configured with AT&T LTE network parameters [9].

Fig. 5.12(a) presents the pairwise energy-delay values of BE next to their corresponding α values. Notice that in this figure, the energy cost is expressed in Joules. Here BE exhibits a behavior similar to the one in simulations as it is able to cover the broad spectrum of the



(a) Energy-delay tradeoff.

(b) Energy savings.

Figure 5.12: LTE experiments using IoT trace.

energy-delay tradeoff. In this experiment, because of the specific power model parameters of AT&T's LTE network, the maximum energy saving and maximum delay reduction are achieved at $\alpha = 9$ and $\alpha = 1800$, respectively. Also, our experiment with the Default algorithm resulted in zero delay and an energy expenditure of 1356.63 Joules. Fig. 5.12(b) plots the energy savings of BE compared to the Default algorithm for different values of α. As the relative importance of delay decreases, higher energy savings are achieved. Across all the values of α, BE can achieve energy savings ranging from 0% to 100%.

Chapter 6

Conclusion

Due to the increasing popularity of applications running on wireless devices, there has been great interest in improving users' quality of experience, specially in terms of energy and delay. However, providing a satisfactory experience for applications operating in cellular networks is challenging. This is due to the unique characteristics of cellular networks and the nature of interactions between applications and their remote servers. Moreover, reducing energy consumption of applications can increase the delay experienced by users, thus reflecting the existence of an energy-delay tradeoff. This has motivated us to design mechanisms that can reduce energy consumption and latency of applications. We also provide a general and systematic solution to balance the tradeoff between energy and delay. The main techniques employed to achieve these goals are speculative loading and request bundling. Various methods such as simulations, live measurement experiments and competitive analysis have been used to evaluate the performance of the proposed solutions.

6.1 Summary of the thesis

- **Reducting latency and energy of mobile web browsing:** In Chapter 3, we proposed a system called WebPro for reducing the latency and energy consumption of mobile web browsing. WebPro is designed to eliminate the initial round-trip time required to discover the list of the objects referenced in a webpage by using a previously recorded resource list of the webpage. Using measurements involving real world websites, we showed that within a few hours, the amount of change in the structure of webpages is relatively low, and hence it is feasible for WebPro to maintain an updated resource list of popular websites. WebPro not only reduces page load time, but also reduces radio energy consumption by

implementing bundling. We performed a detailed set of experiments to assess the efficiency of a prototype implementation of the system. Our results indicate that WebPro outperforms state-of-the-art in terms of the page load time, though the amount of improvement varies between webpages.

- **Balancing energy-delay tradeoff for request bundling on smartphones:** To capture the energy-delay tradeoff that is inherent in request bundling, in Chapter 4, we considered bundling as a cost minimization problem, where the bundling cost is defined as a weighted summation of energy and delay costs. Energy cost was modeled using an On/Off model and delay cost was characterized using two commonly used delay functions, cumulative and max delay functions. We proposed an online algorithm for the problem and obtained provable performance guarantees through competitive analysis of the algorithm. Our algorithm does not make any assumption about the traffic pattern or nature of applications. We then evaluated our algorithm using simulations and live experiments, which showed that, in realistic scenarios, the performance of the proposed algorithm is close to that of the optimal offline algorithm that knows the arrival sequence in advance.

- **Online energy management in IoT applications:** In Chapter 5, we studied the problem of energy management on LTE-enabled IoT devices. Specifically, we considered application message bundling to alleviate the effect of short message transmissions on energy consumption. We formulated message bundling as a cost minimization problem with the same objective function defined in Chapter 4, *i.e.*, a weighted summation of energy and delay costs. However, specific characteristics of the LTE radios were taken into account by incorporating the DRX mechanism in our energy cost model. Based on the new formulation, we developed an online bundling algorithm to solve the cost minimization problem. Our competitive analysis revealed that depending on DRX and application parameters, our algorithm is 1, 2 or 4-competitive with respect to the optimal offline algorithm. We evaluated the performance of the online algorithm using an extensive

set of simulations and real experiment on an IoT testbed. Our results show that, i) in realistic scenarios, our algorithm exhibits a performance better than the one implied by the competitive ratio, ii) depending on application requirements, energy savings ranging from zero to about 100% can be achieved using our algorithm, and iii) ignoring DRX could significantly overestimate or underestimate energy consumption.

6.2 Future directions

- **Design and analysis of a randomized bundling algorithm:** The online optimization problems studied in Chapters 4 and 5 can be viewed as a game played against an adversary. In such a game, the adversary generates the sequence of requests and the bundling algorithm (also referred to as player) makes bundling decisions as requests arrive. In this game, the adversary is aware of the online bundling algorithm employed by the player and can use this knowledge to generate an arrival sequence in a way that maximizes the competitive ratio (CR) of the algorithm. From this standpoint, it is possible that randomizing the behavior of the online player can significantly improve its performance (*i.e.*, CR) [111]. This stems from the fact that by incorporating randomization, the moves of the online player will no longer be certain, which gives less power to adversary in generating such an arrival sequence that maximizes the CR.

  In this context, it is interesting to study the possibility of improving the worst-case performance of our bundling algorithm by designing and analyzing a randomized online algorithm for request bundling.

- **Incorporating energy and delay of radio state promotion in energy cost model:** In our energy cost model considered in Chapters 4 and 5, we only focused on the tail energy in order to characterize the radio energy consumption of a wireless device. However, in 3G and LTE networks, the delay incurred for state promotion and also the amount of energy consumed during state promotion are not negligible [9, 14]. Thus, a more accurate

formulation of the bundling problem can be introduced by incorporating promotion delay and energy in our energy cost model. Notice that in such a system, the wireless device will incur promotion energy and delay only if the idle gap between two consecutive grants is longer than the tail time. In other words, making a grant while the device is in the active state will not introduce any promotion energy. However, the wireless device will suffer from promotion energy and delay if the bundling algorithm decides to make a grant in the idle state. In this context, an interesting problem is the design and analysis of online bundling algorithms that take promotion energy and delay into account.

- **Utilizing statistical information about request arrival process:** In our discussions in Chapters 4 and 5, we assumed the complete lack of knowledge about the timing of future requests. However, in some scenarios it may be conceivable to assume that some limited statistical information about the distribution of request inter-arrival times is known to the online algorithm [112]. Such statistical information can include mean and/or variance of request inter-arrival times. In this setting, it is desirable to investigate the possibility of designing online algorithms that can achieve an improved worst-case performance by exploiting the knowledge of statistical information about the arrival process.

- **Improving the competitive analysis of the Break-Even algorithm:** In Chapter 5, we showed that depending on DRX and application parameters, the Break-Even algorithm achieves a competitive ratio of 1, 2 or 4. However, we did not investigate the tightness of the ratio of 2 in its corresponding regime. Also, we showed that the competitive ratio of 4 is not tight in its corresponding regime. In this regard, it is interesting to investigate the tightness of the competitive ratio of 2 and also examine the possibility of replacing 4 with a better upper bound (in terms of tightness). Moreover, in Chapter 5, we presented the competitive analysis of the Break-Even algorithm only for the max delay function. A possible avenue for future research is to perform the competitive analysis of the Break-Even algorithm when energy cost is modeled based on the behavior of the LTE radio (*i.e.*,

a 3-state energy model) and delay cost is defined as the cumulative delay incurred by all the requests.

Finally, another interesting problem is to study the optimality of the competitive ratios obtained in Chapters 4 and 5. Such a study would investigate whether there exist any deterministic online algorithms whose competitive ratios are smaller than the ones reported in Chapters 4 and 5.

- **Channel-aware request bundling:** In Chapters 4 and 5, we ignored the energy consumed because of bundle transmissions (hereafter called *transmission energy*), and only considered the tail energy in our energy model. However, in a wireless environment, there could be conflicts between reducing tail energy and reducing transmission energy [106]. On the one hand, tail energy can be reduced by delaying data transfer requests and transferring them in bundles. On the other hand, transmission energy can be reduced by granting requests during good channel conditions. Given the impact of both these energies on the total energy consumption of the device, a direction for future research is to consider balancing energy-delay tradeoff in a problem setting where energy cost is defined as the summation of tail energy and transmission energy. The question that needs to be addressed is how to adapt our online bundling algorithm and its analysis to the new energy model.

Bibliography

[1] J. Erman *et al.*, "Towards a SPDY'ier mobile web," in *Proc. ACM CoNEXT*, 2013.

[2] Cisco, "Cisco visual networking index: Global mobile data traffic forecast update, 2016–2021," 2017. [Online]. Available: https://www.cisco.com/c/dam/m/en_in/innovation/enterprise/assets/mobile-white-paper-c11-520862.pdf

[3] Phonearea, "Android's google play beats app store with over 1 million apps, now officially largest," 2013. [Online]. Available: https://www.phonearena.com/news/Androids-Google-Play-beats-App-Store-with-over-1-million-apps-now-officially-largest_id45680

[4] "Bandwidth, proximity, control: Reduce latency to milliseconds," *Nokia solutions and networks white paper*, 2013.

[5] Akamai, "New study reveals the impact of travel site performance on consumers," 2010. [Online]. Available: https://www.akamai.com/us/en/about/news/press/2010-press/new-study-reveals-the-impact-of-travel-site-performance-on-consumers.jsp

[6] Q. Han and D. Cho, "Characterizing the technological evolution of smartphones: Insights from performance benchmarks," in *Proc. International Conference on Electronic Commerce*, 2016.

[7] Ericsson, "cellular networks for massive IoT," 2016. [Online]. Available: https://www.ericsson.com/assets/local/publications/white-papers/wp_iot.pdf

[8] V. Gabale and D. Krishnaswamy, "Mobinsight: On improving the performance of mobile apps in cellular networks," in *Proc. WWW*, 2015.

[9] J. Huang *et al.*, "A close examination of performance and power characteristics of 4G LTE networks," in *Proc. ACM MobiSys*, 2012.

[10] M. Butkiewicz *et al.*, "Klotski: Reprioritizing web content to improve user experience on mobile devices," in *Proc. USENIX NSDI*, 2015.

[11] F. Qian *et al.*, "Profiling resource usage for mobile applications: A cross-layer approach," in *Proc. ACM Mobisys*, 2011.

[12] S. Deng and H. Balakrishnan, "Traffic-aware techniques to reduce 3G/LTE wireless energy consumption," in *Proc. ACM CoNEXT*, 2012.

[13] N. Balasubramanian *et al.*, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Proc. ACM IMC*, 2009.

[14] X. Chen *et al.*, "Smartphone energy drain in the wild: Analysis and implications," in *Proc. ACM SIGMETRICS*, 2015.

[15] F. Qian *et al.*, "Periodic transfers in mobile applications: network-wide origin, impact, and optimization," in *Proc. WWW*, 2012.

[16] L. Xiang *et al.*, "Ready, set, go: Coalesced offloading from mobile devices to the cloud," in *Proc. IEEE INFOCOM*, 2014.

[17] B. Zhao *et al.*, "Energy-aware web browsing on smartphones," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 3, 2015.

[18] L. Wang and J. Manner, "Energy-efficient mobile web in a bundle," *Computer Networks*, vol. 57, no. 17, 2013.

[19] A. Sivakumar *et al.*, "PARCEL: Proxy assisted browsing in cellular networks for energy and latency reduction," in *Proc. ACM CoNEXT*, 2014.

[20] Herrería-Alonso *et al.*, "Adaptive DRX scheme to improve energy efficiency in LTE networks with bounded delay," *IEEE J. Sel. Areas Commun.*, vol. 33, no. 12, 2015.

[21] F. Qian *et al.*, "Web caching on smartphones: ideal vs. reality," in *Proc. ACM MobiSys*, 2012.

[22] F. Qian, "Toward mobile-friendly web browsing," *IEEE Internet Computing*, vol. 19, no. 5, 2015.

[23] B. Han *et al.*, "Metapush: Cellular-friendly server push for HTTP/2," in *Proc. ACM Workshop on All Things Cellular*, 2015.

[24] M. Gupta *et al.*, "Energy impact of emerging mobile internet applications on LTE networks: issues and solutions," *IEEE Commun. Mag.*, vol. 51, no. 2, 2013.

[25] S. Gao *et al.*, "SCoP: Smartphone energy saving by merging push services in fog computing," in *Proc. IEEE/ACM IWQoS*, 2017.

[26] X. S. Wang *et al.*, "Demystifying page load performance with wprof," in *Proc. USENIX NSDI*, 2013.

[27] F. Qian *et al.*, "Characterizing resource usage for mobile web browsing," in *Proc. ACM MobiSys*, 2014.

[28] M. Z. Shafiq *et al.*, "A first look at cellular machine-to-machine traffic: large scale measurement and characterization," in *Proc. ACM SIGMETRICS*, 2012.

[29] X. Wang *et al.*, "Internet of Things session management over LTE—balancing signal load, power, and delay," *IEEE Internet Things J.*, vol. 3, no. 3, 2016.

[30] S.-T. Hong and H. Kim, "QoE-aware computation offloading scheduling to capture energy-latency tradeoff in mobile clouds," in *Proc. IEEE SECON*, 2016.

[31] Nokia, "LTE evolution for IoT connectivity." [Online]. Available: https://resources.ext.nokia.com/asset/200178

[32] R. Ratasuk *et al.*, "Overview of LTE enhancements for cellular IoT," in *Proc. IEEE PIMRC*, 2015.

[33] Rico-Alvarino *et al.*, "An overview of 3GPP enhancements on machine to machine communications," *IEEE Communications Magazine*, vol. 54, no. 6, 2016.

[34] C. S. Bontu and E. Illidge, "DRX mechanism for power saving in LTE," *IEEE Commun. Mag.*, vol. 47, no. 6, 2009.

[35] A. Sehati and M. Ghaderi, "Energy-delay tradeoff for request bundling on smartphones," in *Proc. IEEE INFOCOM*, 2017.

[36] A. R. Karlin *et al.*, "Competitive snoopy caching," *Algorithmica*, vol. 3, no. 1-4, 1988.

[37] A. Sehati and M. Ghaderi, "WebPro: A proxy-based approach for low latency web browsing on mobile devices," in *Proc. IEEE/ACM IWQoS*, 2015.

[38] A. Sehati and M. Ghaderi, "Network assisted latency reduction for mobile web browsing," *Computer Networks*, vol. 106, 2016. [Online]. Available: https://doi.org/10.1016/j.comnet.2016.06.026

[39] A. Sehati and M. Ghaderi, "Online energy management in IoT applications," in *Proc. IEEE INFOCOM (to appear)*, 2018.

[40] M. Varvello *et al.*, "Is the web HTTP/2 yet?" in *Proc. PAM*, 2016.

[41] J. Kurose and K. Ross, *Computer networking: a top-down approach*. Pearson Education, 2012.

[42] B. Thomas *et al.*, "SPDYing up the web," *Communications of the ACM*, vol. 55, no. 12, 2012.

[43] Z. Wang *et al.*, "How far can client-only solutions go for mobile browser speed?" in *Proc. WWW*, 2012.

[44] S. Sundaresan *et al.*, "Measuring and mitigating web performance bottlenecks in broadband access networks," in *Proc. ACM IMC*, 2013.

[45] Z. Wang *et al.*, "Why are web browsers slow on smartphones?" in *Proc. ACM HotMobile*, 2011.

[46] F. Qian *et al.*, "Characterizing radio resource allocation for 3G networks," in *Proc. ACM IMC*, 2010.

[47] S. Rosen *et al.*, "Discovering fine-grained RRC state dynamics and performance impacts in cellular networks," in *Proc. ACM MobiCom*, 2014.

[48] N. Buchbinder and J. S. Naor, "The design of competitive online algorithms via a primal–dual approach," *Foundations and Trends® in Theoretical Computer Science*, vol. 3, no. 2–3, 2009.

[49] S. Albers, "Online algorithms: a survey," *Mathematical Programming*, vol. 97, no. 1-2, 2003.

[50] A. Borodin and R. El-Yaniv, *Online computation and competitive analysis*. cambridge university press, 2005.

[51] D. Lymberopoulos *et al.*, "Pocketweb: Instant web browsing for mobile devices," in *Proc. ACM ASPLOS*, 2012.

[52] Y. Ma *et al.*, "Measurement and analysis of mobile web cache performance," in *Proc. WWW*, 2015.

[53] B. D. Higgins *et al.*, "Informed mobile prefetching," in *Proc. ACM MobiSys*, 2012.

[54] *SPDY: An experimental protocol for a faster web*, accessed December 14, 2015, http://www.chromium.org/spdy/spdy-whitepaper.

[55] *Usage Statistics of SPDY for Websites*, accessed November 27, 2017, http://w3techs.com/technologies/details/ce-spdy/all/all.

[56] D. Stenberg, "HTTP2 explained," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 3, 2014.

[57] J. Khalid *et al.*, "Improving the performance of SPDY for mobile devices," in *Proc. ACM HotMobile (Poster Session)*, 2015.

[58] B. Aggarwal *et al.*, "Stratus: energy-efficient mobile communication using cloud support," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 4, 2010.

[59] L. Wang *et al.*, "Proxies for energy-efficient web access revisited," in *Proc. ACM e-Energy*, 2011.

[60] B. Zhao *et al.*, "Reducing the delay and power consumption of web browsing on smartphones in 3G networks," in *Proc. IEEE ICDCS*, 2011.

[61] *Opera mini browser*, accessed December 14, 2015, http://www.opera.com/mobile.

[62] *Amazon silk browser*, accessed December 14, 2015, http://amazonsilk.wordpress.com/.

[63] R. Chakravorty *et al.*, "Optimizing web delivery over wireless links: design, implementation, and experiences," *IEEE J. Sel. Areas Commun.*, vol. 23, no. 2, 2005.

[64] S. Singh *et al.*, "Flexiweb: Network-aware compaction for accelerating mobile web transfers," in *Proc. ACM MobiCom*, 2015.

[65] A. Sivakumar *et al.*, "Cloud is not a silver bullet: A case study of cloud-based mobile browsing," in *Proc. ACM HotMobile*, 2014.

[66] J.-H. Yeh *et al.*, "Comparative analysis of energy-saving techniques in 3gpp and 3gpp2 systems," *IEEE Trans. Veh. Technol.*, vol. 58, no. 1, 2009.

[67] H. Falaki *et al.*, "A first look at traffic on smartphones," in *Proc. ACM IMC*, 2010.

[68] P. K. Athivarapu *et al.*, "Radiojockey: mining program execution to optimize cellular radio usage," in *Proc. ACM MobiCom*, 2012.

[69] F. Qian *et al.*, "Top: Tail optimization protocol for cellular radio resource allocation," in *Proc. IEEE ICNP*, 2010.

[70] F. R. Dogar *et al.*, "Catnap: exploiting high bandwidth wireless interfaces to save energy for mobile devices," in *Proc. ACM MobiSys*, 2010.

[71] M. A. Hoque *et al.*, "Poster: Extremely parallel resource pre-fetching for energy optimized mobile web browsing," in *Proc. ACM MobiCom*, 2015.

[72] M.-R. Ra *et al.*, "Energy-delay tradeoffs in smartphone applications," in *Proc. ACM MobiSys*, 2010.

[73] F. Mehmeti and T. Spyropoulos, "Is it worth to be patient? analysis and optimization of delayed mobile data offloading," in *Proc. IEEE INFOCOM*, 2014.

[74] A. R. Karlin *et al.*, "Dynamic TCP acknowledgement and other stories about e/(e-1)," in *Proc. ACM STOC*, 2001.

[75] A. R. Karlin *et al.*, "Competitive randomized algorithms for nonuniform problems," *Algorithmica*, vol. 11, no. 6, 1994.

[76] D. R. Dooly *et al.*, "On-line analysis of the TCP acknowledgment delay problem," *Journal of the ACM*, vol. 48, no. 2, 2001.

[77] K. Zhou *et al.*, "LTE/LTE-A discontinuous reception modeling for machine type communications," *IEEE Wireless Commun. Lett.*, vol. 2, no. 1, 2013.

[78] H. Ramazanali and A. Vinel, "Performance evaluation of LTE/LTE-A DRX: A Markovian approach," *IEEE Internet Things J.*, vol. 3, no. 3, 2016.

[79] N. M. Balasubramanya *et al.*, "DRX with quick sleeping: A novel mechanism for energy-efficient IoT using LTE/LTE-A," *IEEE Internet Things J.*, vol. 3, no. 3, 2016.

[80] J.-M. Liang *et al.*, "An energy-efficient sleep scheduling with QoS consideration in 3GPP LTE-advanced networks for Internet of Things," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 3, no. 1, 2013.

[81] *S. Souders. Velocity and the bottom line*, accessed December 14, 2015, http://radar.oreilly.com/2009/07/velocity-making-your-site-fast.html.

[82] J. Huang *et al.*, "Anatomizing application performance differences on smartphones," in *Proc. ACM MobiSys*, 2010.

[83] H. Shen *et al.*, "A proxy-based mobile web browser," in *Proc. ACM Multimedia*, 2010.

[84] *Alexa Internet Inc. "Top Sites in Canada"*, accessed December 14, 2015, http://www.alexa.com/topsites/countries/CA.

[85] A. Gerber, S. Sen, and O. Spatscheck, "A call for more energy-efficient apps," *AT&T Labs Research*, 2011.

[86] R. Mahindra *et al.*, "A practical traffic management system for integrated LTE-WiFi networks," in *Proc. ACM MobiCom*, 2014.

[87] G. Barish and K. Obraczka, "World Wide Web Caching: Trends and Techniques," *IEEE Commun. Mag.*, vol. 38, 2000.

[88] V. Paxson, "Bro: a system for detecting network intruders in real-time," *Computer Networks*, vol. 31, no. 23, 1999.

[89] *Is the web getting faster?*, accessed August 19, 2015, http://analytics.blogspot.ca/2013/04/is-web-getting-faster.html.

[90] G. Cormode and M. Hadjieleftheriou, "Finding the frequent items in streams of data," *Communications of the ACM*, vol. 52, no. 10, 2009.

[91] A. Metwally *et al.*, "Efficient computation of frequent and top-k elements in data streams," in *Database Theory-ICDT*. Springer, 2005.

[92] A. Rao *et al.*, "Using the middle to meddle with mobile," in *Tech. Report NEU-CCS-2013-12-10, CCIS, Northeastern University*, 2013.

[93] F. Qian *et al.*, "How to reduce smartphone traffic volume by 30%?" in *Proc. PAM*, 2013.

[94] L. Deutsch and J. Gailly, "RFC 1950: ZLIB compressed data format specification version 3.3," *IETF*, May 1996.

[95] J. van den Brande and A. Pras, "The costs of web advertisements while mobile browsing," in *Proc. Information and Communication Technologies*. Springer, 2012.

[96] M. Carbone and L. Rizzo, "Dummynet revisited," *ACM SIGCOMM Computer Communication Review*, vol. 40, no. 2, 2010.

[97] R. Netravali *et al.*, "Mahimahi: accurate record-and-replay for http," in *Proc. USENIX Annual Technical Conference*, 2015.

[98] J. Huang *et al.*, "An in-depth study of LTE: effect of network protocol and application behavior on performance," in *Proc. ACM SIGCOMM*, 2013.

[99] B. Han *et al.*, "An anatomy of mobile web performance over multipath TCP," in *Proc. ACM CoNEXT*, 2015.

[100] S. Deng *et al.*, "WiFi, LTE, or both?: Measuring multi-homed wireless internet performance," in *Proc. ACM IMC*, 2014.

[101] W. Wang *et al.*, "Dynamic cloud instance acquisition via IaaS cloud brokerage," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, 2015.

[102] Gartner Report, "Top strategic predictions for 2016 and beyond: The future is a digital thing," *www.gartner.com*, Oct. 2015.

[103] W. Kleiminger *et al.*, "Household occupancy monitoring using electricity meters," in *Proc. ACM UbiComp*, 2015.

[104] P. Gope and T. Hwang, "BSN-Care: A secure IoT-based modern healthcare system using body sensor network," *IEEE Sensors J.*, vol. 16, no. 5, 2016.

[105] U. Kulau *et al.*, "Dynamic sample rate adaptation for long-term IoT sensing applications," in *Proc. IEEE WF-IoT*, 2016.

[106] Y. Cui *et al.*, "Performance-aware energy optimization on mobile devices in cellular network," *IEEE Trans. Mobile Comput.*, vol. 16, no. 4, 2017.

[107] J. Song *et al.*, "EDASH: Energy-aware QoE optimization for adaptive video delivery over LTE networks," in *Proc. IEEE ICCCN*, 2016.

[108] L. Vandenberghe, "Piecewise-linear optimization," *EE236a Lecture Notes, University of California, Los Angeles*. [Online]. Available: http://www.seas.ucla.edu/~vandenbe/ee236a/lectures/pwl.pdf

[109] C. Adjih *et al.*, "FIT IoT-LAB: A large scale open experimental IoT testbed," in *Proc. IEEE WF-IoT*, 2015.

[110] A. Betzler *et al.*, "Experimental evaluation of congestion control for CoAP communications without end-to-end reliability," *Ad Hoc Networks*, vol. 52, 2016.

[111] S. Ben-David *et al.*, "On the power of randomization in on-line algorithms," *Algorithmica*, vol. 11, no. 1, 1994.

[112] A. Khanafer *et al.*, "To rent or to buy in the presence of statistical information: The constrained ski-rental problem," *IEEE/ACM Trans. Netw.*, vol. 23, no. 4, 2015.