Re-architecting Web and Mobile Information Access for Emerging Regions

by

Jay Chen

A dissertation submitted in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

Department of Mathematics

Courant Institute of Mathematical Sciences

New York University

September 2011

Professor Lakshminarayanan Subramanian

Acknowledgments

I would like to start by expressing my deepest gratitude to my advisor, Lakshminarayanan Subramanian (or just "Lakshmi"). It was Lakshmi who set me on the path toward my eventual area of research. Lakshmi has always been generous with his time, and never short on ideas or enthusiasm. Without Lakshmi's courage to pursue the research that inspires him, I would not have found my own passion: to build systems that benefit people - as many people as much as possible by inventing ways to bring technology to people living outside of the privileged regions of the world.

Contributors to this dissertation - This thesis is based on research that I performed over the past five years with many colleagues contributing directly to the work in this dissertation. Many people helped me along the way whose help I could not have done without. The RuralCafe user study would not have been possible without the help of Saleema Amershi and Aditya Dhananjay (Chapter 6.6). Our low bandwidth transport modeling and analysis (Chapter 3.1) was an effort largely attributable to Janardhan Iyengar and long discussions with Bryan Ford. Russell Power implemented the feature reduction algorithm for CIPs (Chapter 7.2.2) in his "spare time". Our ELF deployments (Chapters 2.2 and 5.3) were only possible with help from David Hutchful. Michael Paik conceived of and designed the Epothecary track and trace system (Chapter 11.7). Our work on UjU (Chapter 11) would not have been published without the help of Matt Tierney.

Central to many of the systems developed in this thesis was an exceptional group of Masters students, who I feel privileged to have worked with and mentored over the years. Wei-Chih Lu was central in developing UjU (Chapter 11.3) and pushing it forward in subsequent deployments with Alfredo Hubard and Jesus Garcia Pasquel. Brenan Linn led the development of the early prototype of SMSFind (Chapter 10.3). Trishank Karthik implemented the basic Shark crawler and algorithms that eventually convinced us we needed to do something new for CIPs (Chapter 7.2.4). Faiz Kazi worked on a barcode scanner for low-end phones in UjU (Chapter 11).

My influences - At NYU Jinyang Li and Yaw Nyarko have been exceptional mentors whose unique personal qualities I strive to emulate. Over my time spent at internships at at Microsoft Research and Nokia Research Center, I had the great privilege to collaborate with and be mentored by many fantastic people. I would like to especially thank Eric Brewer, Jonathan Ledlie, Ed Cutrell, William Thies, and Kentaro Toyama for their research mentorship, career ideas, and life advice.

Beyond those mentioned earlier, I would like to thank all of the talented members of our systems and networking group at NYU. I feel especially thankful for the comradarie of Ashlesh Sharma, Nguyen Tran, Arthur Meacham, Robert Soule, Yair Sovran, and Sunandan Chakraborty through the years I spent at the office and in New York.

I would like to thank Alex Snoeren at UCSD for inspiring me in his Distributed Systems course during my Masters program and giving me a chance at research. Also at UCSD, Stefan Savage, Geoff Voelker, and Amin Vahdat were powerful collaborators who taught me and helped develop my interest in pursuing a PhD.

The field work and pilots in this thesis would not have been possible without the help of partners on the ground. There are too many people to name, but I would like to especially thank the folks at Amrita University, Strathmore University, Christel House, Korle-Bu Teaching Hospital, and CAME.

Finally, I would thank my loving and supportive family who have always been there for me. I dedicate this thesis to my mother and father.

Relation to other publications - This dissertation includes prior work by the author. Chapters 2 and 5 subsume prior work on ELF [109]. Chapter 2 and 6 include material that extends prior work on web browsing and RuralCafe [108, 114, 115]. Chapter 7 and 8 include and extend CIP publications [110, 112]. Chapter 10 includes and extends work from SMSFind [111, 113]. Chapter 11 is a combination of our UjU and Epothecary publications [204, 248].

Abstract

A significant fraction of Internet users who reside in low-income regions in developing countries (emerging regions) are subject to very poor network connectivity. In these contexts, the World Wide Web is largely unusable or prohibitively slow. In a similar vein, despite the high penetration of mobile devices, a lack of affordable data connectivity prevents wide-scale adoption of many new mobile information services. This situation is unfortunate because web and mobile technologies can potentially benefit people's lives through economic opportunities, improved efficiencies, and spillover effects.

The conventional models for web and mobile information access are fundamentally ill-suited for emerging regions due to several challenges. The first issue is the content-connectivity gap: network connectivity growth in emerging regions has been relatively sluggish compared to the explosive growth in the size and complexity of web pages. Second, most web browsers spawn 30-50 TCP connections for a single web request which pushes TCP to operate in *sub-packet regimes* and results in a complete break down of TCP congestion control. Third, conventional caching and network-level optimizations yield limited benefits in these settings. Fourth, many mobile users live in rural areas where data connectivity is largely absent (due to low demand) restricting them to only Short Messaging Service (SMS) and voice as the only available communication channels. Finally, economics plays a critical factor in the adoption of new technological solutions; due to low purcasing power, most users use cheap low-end mobile devices with limited functionality.

In this thesis, we have proposed a new web architecture to enable information access across the poor connectivity scenarios commonly found in emerging regions: low bandwidth, intermittent, and mostly disconnected. Our web architecture is highly efficient both in terms of network use and end-user time consumed. Our architecture introduces fundamental changes to several layers of the web stack in order to provide an interactive and useful experience regardless of the network environment.

Through real-world user studies, we begin by characterizing web user behavior in emerging regions and describe a new Markov model to accurately capture TCP behavior in sub-packet regimes. To address the TCP breakdown problem, our architecture introduces an in-network optimization engine (using middle-boxes) to prevent TCP timeouts and ensures that flows make smooth progress despite congestion introduced by pathological sharing. In the absence of middle-box support, we develop a purely end-host web optimization engine (ELF) that improves web page load times through aggressive caching, prefetching, and offline browsing.

At the application layer, our architecture implements three fundamental changes. First, to increase the utility of cached web pages despite intermittent connectivity, we introduce a new vertical caching layer organized around "topics". The vertical caching layer maximizes the utility of cached contents by automatically discovering topics of interest and exposing these topics to the end-users. Second, from the user's perspective, we have designed RuralCafe, an asynchronous web browsing system that decouples system interactivity from underlying network conditions and allows users to seamlessly navigate both live web content (as delivered by the network) and cached content (as delivered by the cache). To satisfy the extreme case of disconnected populations, we propose the design of Contextual Information Portals (CIPs) where the goal is to deliver a vertical slice of the web on specific topics as an offline searchable and browseable repository for users. We have developed CIPs for specific educational subjects based on the school syllabus in Kenya and India and have provided the system as an offline tool for teachers to use web-based resources for enhancing school education.

For mobile information access, we show how to build powerful mobile applications despite extremely constrained data channels such as SMS. This is a challenging research problem since the SMS channel is restricted to only 140 byte messages. We have designed SMSFind, an SMS-based search service that allows a mobile user efficiently search the web using one round of interaction where the search response is restricted to one SMS message. Based on real user queries, we show that SMSFind answers 57.3% of queries answered by a human; in comparison, existing SMS search services such as GoogleSMS answered only 9.3% of the queries. To facilitate the development of general SMS based applications, we developed UjU, a mobile application platform that enables users (with no programming experience) to quickly build efficient and reliable applications. To make efficient use of the underlying SMS channel and minimize operational costs, UjU supports a semantic compression layer that achieves a high compression ratio in comparison to standard compression techniques.

Our work has had real-world impact and almost every system that we have built has also been deployed and used by real-world users. ELF was deployed in a large peri-urban school in India. RuralCafe, vertical caching, and our in-network optimization engine have been deployed and tested by users in Amrita University in South India and other organizations in Kenya. The CIP system is in use by Strathmore University in Kenya to develop syllabus-specific CIPs for nearly 60 Kenyan schools to enhance education. CIPs have also been experimented with by teachers in India for elementary after school programs. SMSFind has been deployed in an open beta in Kenya, and used by inhabitants of the Haruma slum in Nairobi. UjU is used by a teaching hospital in Ghana and a microfinance institution in Mexico for four applications in daily operational use.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Worldwide, the number of Internet users under poor connectivity is exceedingly high. Internet access tends to be slow, unreliable, and largely confined to urban areas. As a result, conventional information access models in emerging regions perform poorly, and the web is largely unusable. Cellular network access is relatively common, but only voice and SMS are available in rural areas. In this thesis, we address these challenges by developing *interactive and usable systems that are network efficient despite extreme infrastructural constraints*. We re-architect existing web and mobile application access models based on a new set of design paradigms centered around providing people in emerging regions with the information they need.

The economic reasons behind the general lack of good connectivity are two-fold. First, the purchasing power in these regions is significantly lower than urban areas. Second, none of the traditional cellular or wire-line connectivity solutions (fiber, broadband and dial-up) are economically viable for rural regions with low user densities [194, 241]. These economic and other social constraints pervade all aspects of technology, and are progressively more prevalent in the physical interior of poor countries.

1.1 The Landscape of Information Access in Emerging Regions

The latest statistics from the International Telecommunications Union (ITU) [30] indicate that in 2010 the number of Internet users was approximately 2 Billion, and the number of Internet users in poor countries was approximately 1.2 Billion [75]. Unfortunately, beyond the high speed frontier of digital infrastructure, unavailable, slow, or unaffordable connections are still the status quo for people despite being defined as "connected". Broadband penetration in poor countries was only 4.4 subscriptions per 100 people compared to 24.6 in rich countries in 2010. Africa, in particular, had broadband penetration rates of less than 1% [75]. Furthermore, the Internet is unusable for a wide variety of infrastructural reasons that result in poor or unreliable connectivity [98].

In contrast to wired Internet, mobile devices are relatively ubiquitous. The estimated mobile penetration rate was 67% in poor countries in 2010, and a worldwide mobile subscription rate was 73% [30]. The high penetration of mobile devices in emerging regions around the world has fueled a wave of development-centric mobile applications and services in the context of health-care, education, microfinance, and e-government. OpenXcode [50], OpenRosa [49], ODK [254], FrontlineSMS [21], RapidSMS [53], OpenMRS [48], and Voxiva [68] are just a few examples of projects that use mobile devices for mobile health. Unfortunately, mobile Internet shares many similar constraints as wired Internet. Cell towers are rarely built in areas with low population density, GRPS is relatively bandwidth constrained, and faster connectivity (e.g. 3G) is typically only available in urban areas. Figure 1.1 illustrates the Airtel network's GSM coverage in Kenya.

Further constraining mobile information access is the prevalence of simple low-cost "feature phones" with limited processing and communication capabilities. Despite the excitement surrounding "smartphones" with increased functionality, 2010 estimates indicate that the ratio of feature phones to smartphones globally is 4 to 1 [45]; in the same year, smartphones comprised

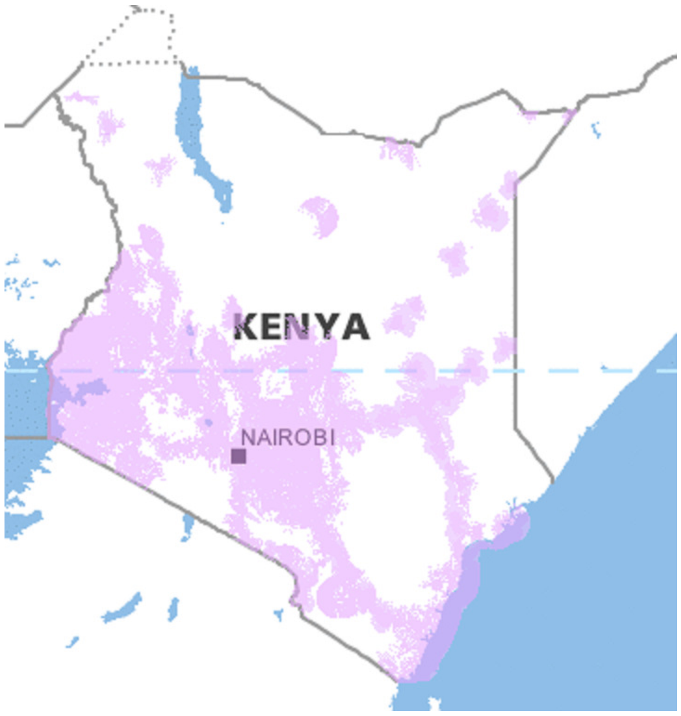


Figure 1.1: Airtel GSM coverage in Kenya 2011 (source: GSM World [29]).

less than 25% of the global handset market [30]. The resulting mobile landscape in emerging regions is a dominance of voice and SMS over data. Reflecting this trend is a range of popular SMS-based applications and services such as mobile money transfer [40], mHealth [21], and social networking [57]. With global SMS volume tripling in three years from 1.8 trillion in 2007 to 6.1 trillion in 2010, SMS will likely continue to remain a primary communications channel in emerging regions.

Of these these Information and Communication Technologies (ICTs), wired Internet and mobile data are pertinent to web access under poor connectivity. Voice and SMS belong to a completely different model of information access.

At a high level, this thesis concentrates on two problems:

1. **Enabling web access under poor connectivity:** We explore the technical reasons why, beyond being slow, web access is broken in emerging regions. We find fundamental issues with the web interaction model, TCP congestion control, and caching. After studying these issues first-hand, we develop systems to solve these problems for their individual connectivity environments. We implement and extensively evaluate our designs and techniques both in the lab and in three pilot deployments in India and Kenya. Finally, we bring our insights together in a new web architecture for enabling web access in emerging regions.
2. **New models of information access for mobile devices:** There are severe problems when using an unreliable, slow, narrow, and expensive messaging system to build applications and services. Most existing applications fail when directly migrated to these settings. Another problem is that operationally critical systems for mobile health, microfinance, and e-government have requirements that are not always met. We show that it is possible to develop relatively powerful services such as web search over the extremely constrained SMS channel. We also design and implement a complete application platform on SMS that enables reliable, cheap, and easy to deploy applications. We demonstrate the effectiveness of our solutions through a large scale pilot of our search engine in Kenya, and the daily operational use of our application platform by a teaching hospital in Ghana and a microfinance institution Mexico.

1.2 Why is Conventional Web Access Broken in Emerging Regions?

The web is an extremely powerful resource that has the potential to improve education, increase efficiencies, and enable access to new markets. There are, however, fundamental problems with web access in emerging regions. The primary issue is that Internet connectivity is not keeping up with web complexity and size. With the leading edge of ICT technology rapidly moving forward and the tail consistently neglected by mainstream technologies, there is no panacea on the horizon. There are several trends that indicate waiting for connectivity to catch up with web growth is not a realistic expectation.

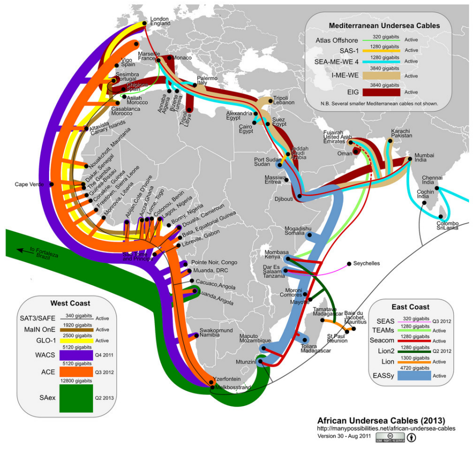
1.2.1 Internet Growth and Penetration are Slow

Akamai quarterly reports consistently indicate that slowest growth is in south Asia, South America, and Africa [6]. Global dialup and broadband penetration growth in emerging regions is isolated to a select few countries on each continent [6, 75]. There is some evidence of this trend changing due to the recent glut of undersea cable projects along the coasts of Africa (e.g. MainOne, GLO1, WACS, ACE, SAex, SEAS, TEAMs, Seacom, Lion, EASSy as shown in Figure 1.2(a)). However, with the physical interior of Africa being difficult to reach, affordable broadband is limited to coastal and high-density urban areas as seen in Figure 1.2(b). The recent emergence of new low-cost connectivity solutions using long-range wireless technologies (WiMax [72], long-distance WiFi [209]) provide hope for rural connectivity, but progress is slow.

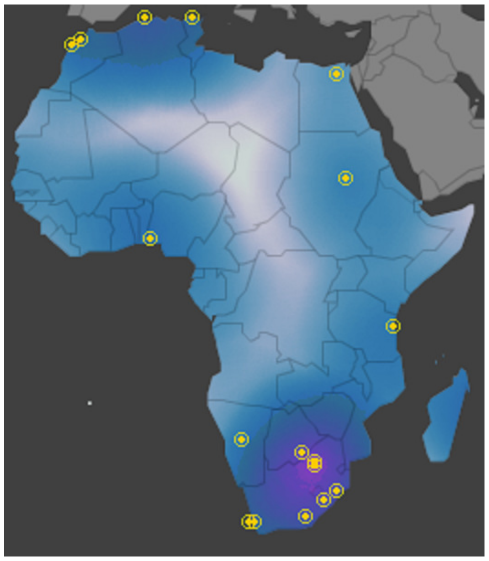
1.2.2 Growth of Web Complexity and Size are Fast

Contrasting the sluggish growth in connectivity are the trends in the evolution of the web itself. Web pages have significantly grown in size and complexity over the past decade. These trends are depicted in Figure 1.3 based on information gathered from several sources [35, 70, 106, 124] indicating a $30 - 50$ fold increase of these metrics in the past decade. The increase in the complexity of the Internet vastly outpaced improvements to connection speed in emerging regions over the same period of time.

In 1995, the average web page size was 14 KB with only 2.3 objects on average; by 2003, these figures grew to 93.7 KB and 25.7 objects [124] and by 2008, to 312 KB and 49.9 objects [70]. In 2010, these figures rose to 507 KB and 64.7 objects for the top 500 pages [106].


Mediterranean Undersea Cables
Atlas Offshore 320 gigabits Active
SAS-1 1280 gigabits Active
SEA-ME-WE 4 1280 gigabits Active
I-ME-WE 3840 gigabits Active
EIG 3840 gigabits Active
N.B. Several smaller Mediterranean cables not shown.
to Fortaleza Brazil
West Coast
SAT3/SAFE 340 gigabits Active
MaIN OnE 1920 gigabits Active
GLO-1 2500 gigabits Active
WACS 5120 gigabits Q4 2011
ACE 5120 gigabits Q3 2012
SAex 12800 gigabits Q2 2013
East Coast
SEAS 320 gigabits Q3 2012
TEAMs 1280 gigabits Active
Seacom 1280 gigabits Active
Lion2 1280 gigabits Q2 2012
Lion 1300 gigabits Active
EASSy 4720 gigabits Active
African Undersea Cables (2013)
http://manypossibilities.net/african-undersea-cables
Version 30 - Aug 2011

(a) Undersea cables connecting Africa (planned to 2013) [3].



(b) High-level broadband speed in Africa as observed by Akamai (2011) [5]. (Broadband = 2 Mbps, darker colors indicate higher speed broadband.)

Figure 1.2: Broadband access in Africa.

In 2010, Google conducted a large scale crawl of the web and found the average web page size across 4.2 billion pages to be 320 KB in 2010 [35].

The contrast between available connectivity and web growth is alarming. Despite increasing the bandwidth available in poor countries, unless the rate of bandwidth expansion per capita increases faster than the rate of web complexity, web access in emerging regions will actually become *worse* relative to today.

1.2.3 TCP Breaks Down

Web pages frequently contain content loaded from several different servers. Standard browsers have evolved to optimize for opening multiple TCP connections (4-6) to each web server yielding

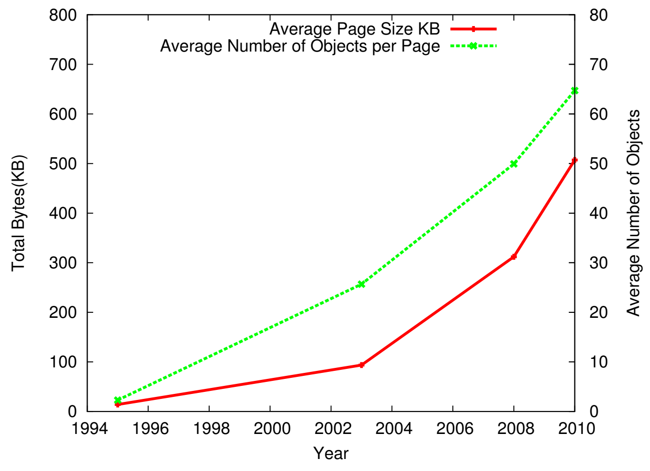


Figure 1.3: Web growth over time.

many competing TCP connections to download one page. Firefox recently increased its default maximum number of connections per server from 4 to 6 and allows 30 TCP connections in total. Users of high bandwidth connections will not notice any negative effects, but if there is enough congestion, users in low bandwidth environments will experience TCP collapse.

Figure 1.4 illustrates the effects of TCP collapse on object download times. The Y-axis spread in the graphs is striking: download times for objects vary by over two orders of magnitude! A significant number of users experience poor performance with large download times even for very small object sizes where the entire object can be transmitted in a few packets. Furthermore, the high variation in the download times for comparable object sizes indicates a high level of unfairness across flows. The resulting effects on user experience for web browsing include unpredictable performance, frequent connection hangs, and malformed web pages. Unfortunately, network conditions with extremely high sharing such as these are commonplace in emerging regions.

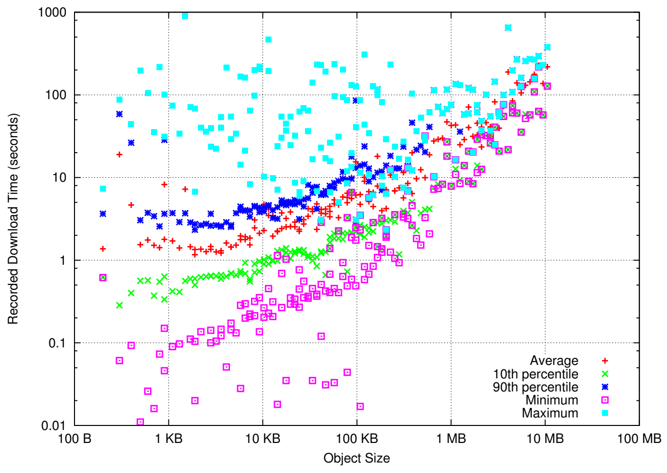


Figure 1.4: Scatter-plot of download times for different object sizes, taken from a 2-hour observation period at a highly congested university Squid proxy. Each raw data point is assigned to a bucket, and the values shown here are the 10th percentile, 90th percentile, min, max, and average values per bucket.

1.2.4 Conventional Caching is Not Enough

The increasing complexity and dynamic nature of web pages presents challenges to caching as well. When a user requests a web page, a series of DNS lookups are made to locate the server hosting the page. Then, a connection is made to the appropriate server and the content is downloaded. The web browser parses the page for embedded links and scripts, and downloads those from their appropriate servers. The scripts may execute and request further resources from yet another server and so on. This web access model works well for good connections, but in high latency or intermittent networks each request over the network results in a substantial delay in the user experience. DNS and content caching by local and intermediate servers or proxies may substantially cut down on the number of network requests. However, if a single cache miss results in a stall on the order of minutes or even hours, it is unlikely that the granularity of caching should remain only at the level of individual URLs.

The current web access models (ASP.NET, Java, PHP, and HTML5) put caching options in the hands of web developers. One available tool for web developers to avoid caching problems is the use of manifests. Web application languages (and HTML5) allow manifests to be defined that indicate the complete set of resources required for offline web applications and pages to be presented correctly. Manifests do mitigate cache misses for loading an individual page, but this technique does not prevent cache misses upon further user requests of hyperlinked pages. Furthermore, web pages are far from universally implementing manifests.

On the client end, more aggressive caching should intuitively be employed in more constrained network conditions. However, existing web standards are optimized for web delivery over high speed connections and are not designed for extremely constrained networks. Alternatively, proxies-based solutions require expertise to deploy and relatively extensive effort to manually optimize for challenged networks. Finally, browser caching options are limited by existing web standards and by similar inattention to design for constrained environments.

1.2.5 Data Connectivity Will Cost Money

Cellular data will likely continue to increase in speed in urban areas, but fail to extend to rural regions due to economics. Even if data access becomes available in emerging regions, it remains unclear whether it will be affordable for rural users. Mobile operators in poor countries are typically “pre-paid” and charge per MB used. Despite presumably continued price drops, bandwidth will remain a consideration for any phone based application. Even mobile operators in rich countries have recently shifted toward “tiered” pricing plans rather than unlimited data. In the United States, AT&T began tiered pricing in June 2010, T-Mobile in June 2011, and Verizon in July 2011 [62].[1]

[1] At the time of this dissertation only Sprint Nextel remains as a Tier 1 operator without tiered pricing in the United States [1].

1.2.6 Rewriting the Web is Not Scalable

One approach to help scale down the web for mobile devices has been to rewrite pages for mobile phones. This has been relatively successful in the major websites that can afford to design and support two versions of their websites. A similar solution may benefit hosts behind constrained wired networks. However, the general approach (i.e. manually rewriting and optimizing all web pages for constrained networks) is not scalable in its current form.

Browsers that automatically adapt web pages to device and network constraints are a promising direction. Opera Mini [51] which pre-renders pages using proxy servers employs only conservative optimizations that are (appropriately) limited to mobile web.[2] Loband [37] is a system and a website that converts web pages to text-only, but is even more limited in its effectiveness for network constraints beyond bandwidth. These systems for web adaptation show that page adaptation is a good alternative to manual page rewriting, but there is definitely room for a systematic and generalizable approach to web content adaptation.

1.3 Understanding Web Access in Emerging Regions

There is a lack of understanding in the networking community on how networking technologies are being used by people in emerging regions, where the failures are, and how the situation may be improved [251]. There are two central reasons for this knowledge deficiency. First, emerging regions are extremely heterogeneous; there are huge infastructural and social differences between an after school program in a small town in South Asia (Figure 1.5(a)) and a slightly rural secondary school in Africa (Figure 1.5(b)). The root causes of poor connectivity are varied, as are the resulting technical challenges. Researchers have analyzed web traces from emerging regions before [125, 146, 150, 157, 200], but these studies explore only a limited set of network

[2]As of 2011, Opera Mini is being used by 72% of mobile web users in Africa [60].

conditions. These few research efforts are meager compared to the vast diversity of contexts and challenges. Second, in the mainstream networking literature, Internet access studies are outdated and have gone out of vogue in recent years [180, 202] in favor of studies on mobile use [160, 252, 255].

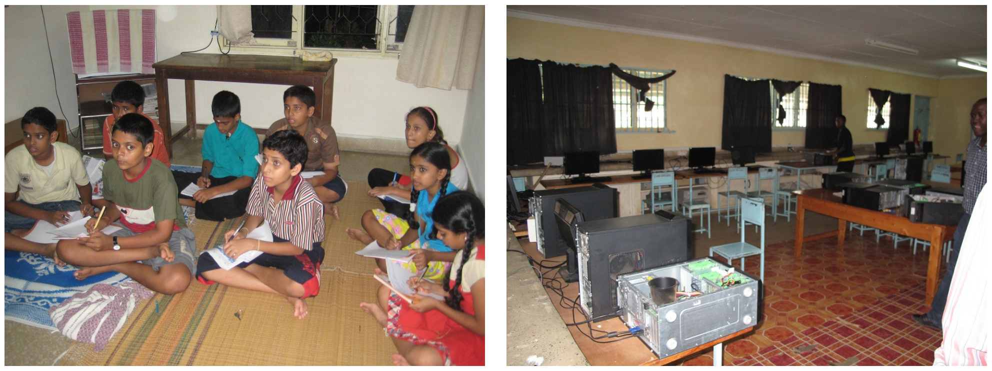

(a) After school program in Kalpakkam, India. (b) Computer lab outside Nairobi, Kenya.

Figure 1.5: Two completely different emerging region contexts.

Outside the networking community, development centric researchers do study Internet use, but these studies are rarely conducted at a level that produces insights useful to networking researchers [146, 251]. This is unfortunate as the case studies themselves often identify real problems with access, but salient technical issues are not explored in depth. To address these knowledge gaps, we begin with two chapters studying the networking challenges in different archetypically poor networking environments.

Chapter 2 - Web Use in Emerging Regions. In this chapter we conduct a pair of studies to understand how people currently access the web under poor connectivity.

The first study is an analysis of web access patterns at a school outside Bangalore, India. The infrastructure at this school is good relative to rural areas, and represents one of the least infrastructure constrained scenarios on the spectrum of poor connectivity. This study focuses on characterizing web use in emerging regions and providing a tool to study access patterns at scale. The data collection tool we developed is bundled as a Firefox plugin that is immediately accessible to non-expert users for both installation and use. Our tool is is highly appropriate for use by other researchers due to the low requisite expertise, software, and expense. Our tool has been used in two other studies [47, 222] since it was open sourced in February 2011.

The second study was conducted at a university in rural Kerala, India. The university is equipped with a 8 Mbps link, but the per-machine fair share of the bandwidth is only 1.9 Kbps. As a result, the connectivity experienced by the user is slow and sometimes intermittent. This study focuses on user behavior rather than access patterns.

Together, these two studies are illustrative of how the web is used in emerging regions, how conventional web optimizations perform (and break down) in these environments, and how to improve the status quo.

Chapter 3 - The TCP Breakdown Problem. In this chapter we thoroughly explore a pathologically shared network connection at the transport level and illustrate why existing congestion control schemes such as TCP-NewReno [134], TFRC [143] and many others fail. We define such environments, where a TCP flow's fair share is less than 1 packet per RTT, as the *sub-packet regime*.

We make several contributions towards characterizing TCP behavior in sub-packet regimes. By analyzing the per-flow and aggregate behavior of TCP and other variants in the sub-packet regime, we show that apart from the well-known problems of high loss rates and poor performance, flows experience the following: (a) extreme unfairness over the short time scale; (b) arbitrary admission control; and (c) repetitive timeouts that force a significant fraction of flows to observe long silence periods with no packet transmissions. In addition, we show that none of the standard TCP variants or known queuing mechanisms offer substantial performance gains in the sub-packet regime.

To better understand this phenomenon, we introduce an analytical model to characterize the equilibrium behavior of TCP in the sub-packet regime. Our model is a simpler variant of a full Markov model for TCP operating in traditional regimes [136], but with an emphasis on modeling repetitive timeouts.

1.4 A New Web Architecture for Emerging Regions

The evidence indicates a need to fundamentally rethink the model of web delivery in emerging regions. In this thesis, we propose the design of a web architecture that preserves a good user experience despite the underlying network conditions of different contexts. The design of our architecture revolves around two main ideas: *decouple web browsing interactivity from the underlying network*, and *expose the condition of the network to users in an actionable manner*. Figure 1.6 illustrates the components of our proposed web architecture in two different deployment scenarios. The first scenario is with modifications solely at the end-host. The second scenario is in the case of middle-box support (e.g. we are allowed a well-connected host across the constrained network link).

Low bandwidth transport layer: For low bandwidth connections with high levels of sharing, TCP and nearly all of its variants break down. To maximize transport performance, our system supports an *in-network optimization engine* that uses a combination of flow aggregation, admission control, fair scheduling, and fine-grained packet prioritization. These mechanisms keep TCP operating as intended despite congestion introduced by pathological sharing while requiring no software modifications at the end-hosts.

End-host Optimizations: We apply conventional end-host web optimizations that may be implemented with minimal intrusiveness. These optimizations include modifications to caching, prefetching, and offline mode behavior in a standard web browser. Some modifications do vi-

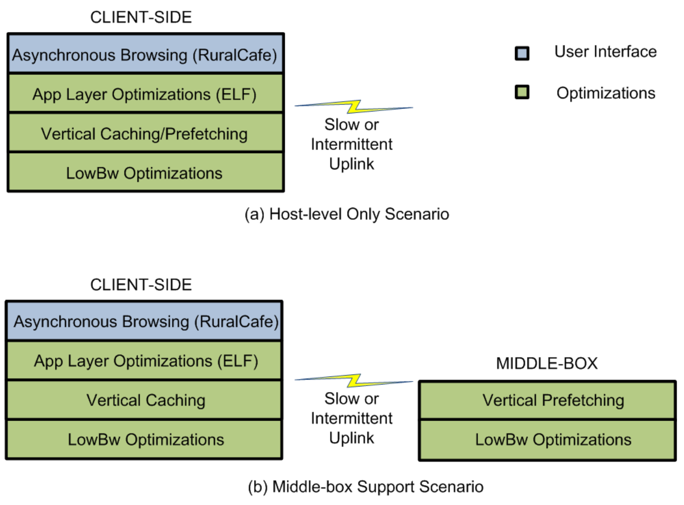

CLIENT-SIDE
Asynchronous Browsing (RuralCafe)
App Layer Optimizations (ELF)
Vertical Caching/Prefetching
LowBw Optimizations
Slow or Intermittent Uplink
User Interface
Optimizations
(a) Host-level Only Scenario
CLIENT-SIDE
Asynchronous Browsing (RuralCafe)
App Layer Optimizations (ELF)
Vertical Caching
LowBw Optimizations
Slow or Intermittent Uplink
MIDDLE-BOX
Vertical Prefetching
LowBw Optimizations
(b) Middle-box Support Scenario

Figure 1.6: Web architecture components.

olate existing web standards, but unlike the other components of our architecture this set of optimizations may be deployed immediately with no expertise or support from proxies and does not significantly alter the current browsing model.

Asynchronous web search and browsing: Through the use of proxies our architecture provides an intermittency-aware web browsing model that allows users to navigate both live web content (as delivered by the network) and cached content (as delivered by the cache). Furthermore, the proxies provide users with control over the limited available network resources.

Vertical caching: We introduce a new concept called vertical caching to maximize the utility of cached contents. The idea of vertical caching is that despite encountering a cache miss, the cache may still contain the *information* desired by the user. Thus, unlike conventional caches that are solely URL based, our cache is also organized based on *topics*. These topics are exposed to users allowing direct interaction and exploration of cache contents. Prefetching is similarly topic-based.

In this thesis we explore a wide variety of constrained network conditions in emerging regions and develop a set of systems designed to optimize for individual problems. Our insights and techniques are gathered and reconstituted into a complete architecture design in Chapter 9. Our general architecture may also be applied to mobile web access, which shares similar network constraints.

1.5 Realizing the New Web Architecture

Our system design has evolved over a 4-year period where we have deployed variants and components of the whole under different poor connectivity conditions as described in the previous chapters. The lessons learned from building and deploying these systems are brought together in a cohesive web architecture that is a significant departure from the existing model. We outline the projects that led to components of our architecture and our contributions below.

Chapter 4 - Fixing the TCP Breakdown Problem: A Middle-box Approach. In this chapter we explore ways to apply our model toward a solution to the TCP breakdown problem. We design and implement an in-network solution to avoiding TCP breakdown under high sharing, and demonstrate its effectiveness in improving fairness and reducing object download time. Our solution extends the good operating range of TCP, and keeps TCP within this range despite pathological link sharing.

Chapter 5 - ELF: An End-host Web Optimization Engine. Our emphasis in this chapter is on deploying a practical tool to accelerate web traffic in the school environment. This tool leverages techniques for aggressive caching, including serving stale pages, client-side prefetching, and other proposed but incompletely evaluated web optimizations.

We implement our tool as a Firefox extension bundled with ELF, our data collection extension, and formally evaluate these web optimizations at a school in peri-urban India. We find that in a slightly congested environment, our tool can provide 2.2x average speedup of page loads, but naive client-side prefetching is largely ineffectual.

Chapter 6 - RuralCafe: An Asynchronous Model for Web Search and Browsing. In this chapter we design RuralCafe, a web search and browsing system that uses an asynchronous queueing model. RuralCafe decouples the user interaction from underlying network conditions. RuralCafe attempts to maximize the utility of network connectivity by giving users actionable information about the status of the network.

We conduct an extensive user study at a university in rural India to explore how browsing behavior is affected by an asynchronous queueing model. We find that overall, an asynchronous queueing model of web browsing is at least as good as conventional browsing in a highly congested network environment.

Chapter 7 - CIP: A Vertical Web Access Layer for Disconnected Web. In this chapter we design and implement a system to automatically construct a *Contextual Information Portal (CIP)*. A CIP provides an offline searchable, browsable repository of content from the web about specific topics. We describe the design and implementation of a complete system to automatically generate CIPs for any topic that requires: (a) a document classification engine that can determine with high accuracy whether a page is relevant to a topic or not with minimal training; (b) a focused web crawler that can efficiently crawl the web starting from a few authoritative pages to determine the set of relevant pages on a given topic.

We evaluate our system across several metrics: classification accuracy, crawl scalability, crawl accuracy and harvest rate. We describe the utility and usability of our system based on a preliminary deployment study at an after-school program in India, and also outline our ongoing larger-scale pilot deployment at five secondary schools in Kenya.

Chapter 8 - A Vertical Caching Layer for Low Bandwidth Networks. This chapter extends the CIP concept to caching for low bandwidth networks. To maximize the utility of cached contents we alter the definition of a cache hit to include pages that contain the relevant content. We introduce a new cost metric for cached objects to accurately capture object cost as a function of time spent downloading and show that a combination of different types of topics can effectively identify and capture a large fraction of requests in small communities such as schools.

Chapter 9 - A Web Architecture for Emerging Regions. In this chapter, we argue that there is a need to fundamentally reconsider the model of web delivery in emerging regions. Inherent disjunctions between existing standards and what works in highly constrained networks appear constantly across the networking stack. We design an end-to-end system that enables web access in emerging regions under all types of poor connectivity. Our system is designed to provide an interactive web experience for poor network connections across the gamut of network environments present in emerging regions.

1.6 Rethinking Mobile Information Access

In the last two chapters of this thesis we discuss new models for information access constrained network channels. We focus on SMS as the communications medium as it is the most ubiquitous and simultaneously the most constrained. Our techniques show that it is possible to develop powerful and reliable applications even in the most constrained network conditions.

One of the most common and powerful web services is *search*. Unlike desktop web search, users on mobile devices rarely have the luxury of iteratively refining search queries or sifting through pages of results for the information they want [237]. Search over SMS is even further constrained due to the 140-byte SMS size, delay, and small form factor of mobiles. Despite these limitations, SMS-based search is a rapidly growing global market with over 12 million subscribers as of July 2008 [119].

To mitigate the difficulties with SMS-based search, existing automated services such as Google

SMS [26] and Yahoo! oneSearch [77] encourage users to enter queries for a number of pre-defined topics, or *verticals*. These pre-defined topics are either identified through the use of special keywords within the search query such as “define” or “movies” (e.g. Google SMS: “define boils”) or have specialized parsers to determine which of the topics is intended (e.g. querying “AAPL” to Yahoo! oneSearch is a query for information about “stock quote”). ChaCha [13], a SMS-based question answering system, hires humans to search the web and answer questions in an SMS response.

Although pre-defined topics do cover a large subset of search requests, none of the existing automated SMS search services is a complete solution for search queries across *arbitrary* topics. Similar to traditional web search queries, SMS search queries suffer from the *long tail phenomenon*: there exists a long tail of search queries whose topics are not popular and typically require a human to answer (e.g. “what are graduation gift ideas?” or “what chemicals are in a fire extinguisher”). In our sample of ChaCha questions only 21% of queries belong to verticals while 79% are long tailed.

Chapter 10 - SMSFind: A New Mobile Search Service. In this chapter, we address the problem of *SMS-based search*: *how does a mobile user efficiently search the web using one round of interaction where the search response is restricted to one SMS message?*

We describe the design and implementation of SMSFind, an SMS-based search engine specifically designed for the long tail of search queries that are spread across a wide range of topics. We evaluate SMSFind on a set of queries from ChaCha [13], and show that SMSFind can answer 57.3% of queries answered by a human. We then improved our system through feedback from focus groups and user studies in Haruma, an urban slum, in Nairobi, Kenya. Finally, we released SMSFind in an open beta in Nairobi.

One fundamental roadblock to large scale adoption of SMS-based *applications* (e.g. medical record systems, data collection applications, etc.) has been that of *operational sustainability*.

While the costs of deploying SMS applications is manageable for pilots, the operating expense of large scale SMS-based systems may actually eclipse the value of the projects they are intended to support. SMS applications like FrontlineSMS [21] and RapidSMS [53] are designed for rural settings and require only low-end mobile devices, but the messaging systems utilize the limited bandwidth inefficiently and the applications themselves are feature poor.

Chapter 11 - UjU: Redefining the Application Stack for Mobile Services in Emerging Regions. In this chapter we present the design and implementation of UjU, a mobile application stack that enables users to develop efficient and reliable SMS-based mobile applications quickly and easily. The design of UjU makes three important contributions as illustrated in Figure 1.7. First, UjU significantly simplifies the design of new SMS based applications on top of a common platform. Users with no programming experience can potentially build new SMS based applications quickly. Second, to minimize the operational cost of SMS-based applications, UjU supports semantic compression and aggregation layers that minimize the number of messages sent over the air. Third, UjU provides a lightweight reliability layer to handle message losses or reordering effects of the SMS channel.

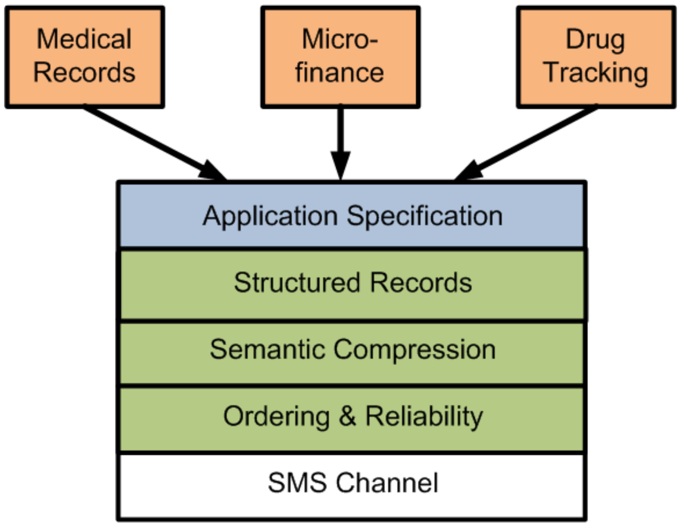


Figure 1.7: The UjU design stack.

We demonstrate the utility of UjU using four real-world applications: (i) a pharmacovigilance system deployed in Ghana; (ii) a mobile microfinance application deployed in Mexico; (iii) a customer service application for microfinance customers deployed in Mexico; (iv) an essential drug list query application for doctors, health workers and pharmacists used in Ghana.

1.7 Impact

Multiple user studies were conducted throughout the course of system design and prototyping. Five complete systems were implemented and deployed in the field. While the impetus of this thesis is solving networking problems, the ultimate purpose of the research in this direction is to promote sustainable development. Based on this, the most meaningful metric for success is positive impact. We believe that the systems we built demonstrate positive impact and summarize some of our notable results here.

End-host Web Optimization Engine (ELF) - We designed our web optimization engine, ELF, to be easy to use, scalable, and to require little expertise or permissions. These properties are particularly crucial due to the challenges involved in conducting field research in emerging regions. We have deployed ELF at an elementary school in Bangalore, India. The success of ELF may be measured by the fact that since its release in February 2011, it has already been used in two other studies and slowly augmented [47, 222]. We hope that this adoption continues, and we will support the ongoing development of our tool.

Asynchronous Web (RuralCafe) - RuralCafe was designed for intermittent environments and as an extensible platform for any constrained network situation. Since the initial development of RuralCafe in 2008, RuralCafe has been used extensively by a university in rural Kerala, India to optimize web access. Components of RuralCafe were also as a part of our education portal projects to provide an offline web experience. The code base for RuralCafe has been open

Figure 1.8: ELF deployment site, a peri-urban elementary school lab in Bangalore, India.

sourced and is used as a teaching platform by university students to teach computer science to secondary school students in Kenya (Figure 1.9).

Figure 1.9: RuralCafe deployment site, a secondary school lab in Nairobi, Kenya.

Contextual Information Portals (CIPs) - We developed CIPs for an after school program in Kalpakkam, India and five schools near Nairobi, Kenya (Figure 7.3). We have had some extremely positive feedback from some teachers and students:

"The students love it! How can we get more contents?

and

"I am using it on weekends. The other teachers want to use it too. I am embarassed to ask. I want more contents for only my class."

Following our pilot deployments, we have been working closely with our collaborators on the ground who organized a local educator's conference with representatives from over 60 secondary schools in attendance. At this conference we showcased CIPs and assessed interest in our technology. We received extremely positive feedback and are working with these schools to scale up our pilot and solicit participation for a larger scale formal study emphasizing impact on teacher self-efficacy and technology skills. Also, in Ghana, we have been actively working on a large scale and long term evaluation focusing on improvement in student educational outcomes.

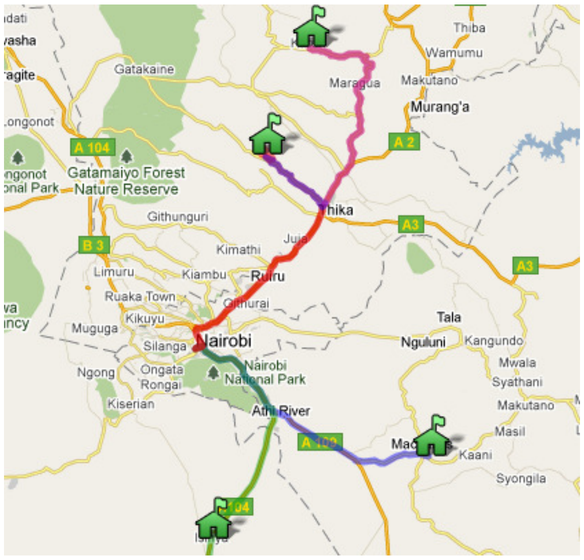


Figure 1.10: CIP deployment sites at five secondary school outside Nairobi, Kenya.

SMS-based Search (SMSFind) - We worked with people living in Haruma slum in Nairobi, Kenya to assess the need for and design of SMSFind (Figure 1.11). Our slum participants found the SMS search capabilities extremely beneficial for searching for quick facts and local information. One participant commented:

> *"If I am looking for a company, I can use this to find where it is."*

Our study participants cited the cost of sending an SMS as a concern, (the price point at the time of the need assessment study was approximately 0.06 USD per message in February 2010), but SMS costs in Nairobi have since dropped to 0.01 USD per message as of July 2011.

After our focus group and design iterations, we released SMSFind in an open beta in Nairobi, Kenya in Summer 2010 where it was used by the public for 2 months.

Figure 1.11: SMSFind users, Haruma slum inhabitants in Nairobi, Kenya.

SMS-based Application Platform (UjU) - We developed several applications for a teaching hospital in Ghana including a pharmacovigilance system and a drug list application. In Mexico we developed a mobile microfinance application and a customer service application in daily operational use. We are in the process of developing a drug track and trace system as a part of an effort to eradicate counterfeit drugs in emerging regions (Figure 1.12).

Today, powerful ICTs such as web and mobile applications are not accessible to low-income users in developing countries because the vast majority of technological innovation is designed for and affordable by rich people in rich countries. In this thesis, we attempt to reverse this trend by focusing on appropriate design for the "next billion" people in emerging regions. We re-architect existing web and mobile technologies by following a set of new design paradigms emphasizing availability, usability, and bandwidth efficiency. We believe that our solutions show how to significantly lower the requirements of technology to make them accessible to people who have traditionally been underserved by technological progress. Specifically, we contribute numerous observations about information access in emerging regions, and explore why current models for web and mobile information access are broken. While we develop new techniques and systems to solve these problems, we consider one of the main contributions of this thesis to be breaking ground and defining problems to enable future researchers to enter this area.

Figure 1.12: Pharmacy for assessing Epothecary deployment in Rongai, Kenya.

Chapter 2

Web Use in Emerging Regions

In this chapter, we conduct two independent studies to understand web use in emerging regions.

In our first study, we analyze six weeks of HTTP traffic from a primary school outside of Bangalore, India. We characterize the information needs and web content requested by students and staff behind a fairly congested 2 Mbps Internet connection shared by 97 computers. Compared to previous studies, we find a shift of requested web content towards more dynamic pages and a dominance of video and advertising content, even in the absence of a very high-bandwidth connection.

In our second study, we observe university students and staff browsing from behind a heavily congested 8 Mbps bottleneck connection shared by 400 computers on their campus in Kerala, India. Here, we focus on understanding user *behavior* and the user experience. We observe tremendous fluctuations and apparent outages of the network that resulted in extreme user frustration and wasted time. We also find that existing web optimizations fail to preserve an acceptable user experience.

2.1 Motivation

Relatively little is known about web use in emerging regions particularly among the poorly connected [251]. Research that focuses on traditional constrained web access is either outdated [180, 202] or, in the case of more recent work, is in the context of mobile devices [160,252,255]. Characterizing traffic patterns of groups of users in emerging regions has only been performed in a few isolated cases at Internet cafes or kiosks in rural areas [125, 146, 157, 200]. This dearth of information is largely due to the same challenges that inhibit connecting these areas in the first place [98]. One recent attempt to gather web traffic from emerging regions at larger scales uses Content Distribution Networks (CDNs) [148]. However, collecting traffic statistics from client opt-in CDNs biases the sample population since the traffic traces themselves only contain requests that use the CDN. The difficulties in collecting traffic data indicates a need for better monitoring tools.

The lack of adequate basic research on web traffic in these underserved regions is somewhat expected, but a large and growing number of initiatives, systems, applications, and services are targeted at emerging regions each year [91, 122, 223, 224]. Web use in schools is particularly important to study. Education is a central focus of development initiatives [66]. Also, digital literacy is increasingly considered integral to getting a good job in countries such as India, Kenya, and Ghana. Internet access is viewed as a bootstrapping mechanism for both education and e-literacy. Because of this, millions of dollars are poured into connecting schools in emerging regions to the Internet, with donations from governments, international organizations, non-profits, and private sponsors. Without a basic understanding of the needs of the target population many efforts are at best misguided and at worst a waste of resources that could be better spent elsewhere.

In this work we study web use in emerging regions in the important context of education. Our studies complement previous studies on web use in telecenters, Internet cafes and kiosks [125, 146, 157, 200]. We conduct two independent studies using different methodologies and in different contexts to explore web use in detail. While similar studies have been conducted previously, we are unaware of comparable deployments in similar contexts.

Our first study is at a primary school in peri-urban India outside of Bangalore. We conduct a set of informal surveys and develop a data collection tool to analyze HTTP traffic through the school. One novel aspect of our data collection tool is that it is bundled as an open-source Firefox plugin and runs directly on client machines. This allows easy installation and configuration by end users, and is especially important given the many constraints on security, privacy, and local technical expertise in our target environments. A simple browser extension can enable incremental and non-intrusive deployments that minimize system initial and operational overheads.

Our second study is at a rural university in Kerala, India. In this study we concentrate on user browsing behavior under poor network conditions. We observe extremely large disparities between expert and novice web users in time wasted due to the network. Furthermore, the existing caching mechanisms were completely inadequate in this more constrained setting resulting in a frustrating user experience. The differences between the two studies conducted in the same country highlights the diversity of networking conditions in emerging regions.

Our studies provide a clear picture of web use in two common contexts in emerging regions, and motivate research into improving web access in emerging regions.

2.2 Study 1: Peri-Urban Primary School in India

This study was conducted in a primary school outside of Bangalore, India. We characterize the environment as “peri-urban” because it is situated amongst farmland about 12 miles north of the city center, translating to about a one-hour drive in typical conditions. While the quality of the teachers and the facilities are above typical Indian standards, all the students come from households that earn at or below the poverty level. The school is a kindergarten through 10th-grade institution with 922 students and 51 teachers. As a school that is close to one of the most well-connected cities in India, access to Internet connectivity is relatively good. However, it is still susceptible to unplanned loss of power and Internet connectivity. This is mostly due to either nearby construction work or infrastructural deficiencies on the service provider's end. In such situations, power from UPS systems is made available to the computers of a few important staff members who then access the Internet via satellite data cards.

Under normal operation, the school shares a 2 Mbps link between 97 computers all running Microsoft Windows XP. Of the 97 computers, students have access to 2 labs running 58 machines; the teachers share 11 computers situated in their lounge room; and the rest of the computers are used by administrative staff. All Internet traffic goes through a gateway machine kept in a secure room on the school premises. Only the technology vendor for the school has access to the gateway machine. The Internet connection costs USD 180.00 a month for an upload bandwidth of 0.34 Mbps and download of 0.85 Mbps. Additional usage costs USD 0.004 per megabyte. The typical use rate for this school is 8 gigabytes downstream out of an available 10 gigabytes per month.

2.2.1 Internet Usage

Prior to the experiment, we conducted a survey to determine the nature of Internet usage at the school. We limited our study to the senior computer lab, which has 40 machines and serves 5th-10th grade students, and the teachers' lounge, which has 11 computers. Previous observations indicated that these were the most active users of the Internet. There were 30 participants in the survey (11 teachers and 19 students). From the survey results, the sampled students reported using the Internet only 1-4 times per week, on average. This was different than the teachers, who reported daily usage. Both students and teachers reported that, on average, each online session lasted less than an hour.

This means that the students only use computers and access the Internet when they are at school and only during specified computer class times. Only senior students can have unsupervised access to the computers outside scheduled class sessions. We hypothesized that this time-limited and structured access to the Internet would influence the sites visited by the students. This was supported by the survey results where the students reported frequently visiting web-based email sites and accessing curriculum-related websites, but rarely blogs, news, shopping and social networking websites, which are otherwise popular among children their age. The teachers also frequented web-based email and education sites more often than the other types of sites. Further interviews revealed that the high webmail use was a result of a school-wide policy for teachers to share their weekly lesson plans with their heads of departments only through email. The policy was in place to encourage teachers to learn and use the Internet. Both students and teachers also reported visiting video sites – this was the third highest reported destination.

When queried about their response to slow connectivity, both teachers and students indicated that they got frustrated and considered it to be a waste of time waiting for a page to load. In such situations, they reported that they coped by loading only one page at a time. Only one person remarked that they used the browser's features, such as disabling images or javascript, to speed up the page load. When asked about offline browsing, none of the respondents reported having used it.

Although the observations and survey focused on the most active Internet users in the school, we believe it to be representative for the rest of the school, and to a large extent, other schools in the area. Ultimately, understanding the costs of connectivity, the patterns of use and the nature of the browsed content provides us with a more informed lens to analyze and improve web access in these types of environments.

2.2.2 Methodology

In order to analyze and accelerate web traffic in the school environment, we implemented a custom tool: the Event Logger for Firefox (ELF). ELF is a free and open source plugin for Firefox that enables logging of Internet traffic The complete ELF extension was implemented in approximately 2,000 lines of code (including comments and blank lines). We describe the requirements and capabilities of this tool in the remainder of this section.

We initially intended to install a data collection mechanism at the school gateway proxy or simply copy the logs from that machine for analysis. After some discussion with the administration we discovered that the network maintenance was being outsourced to an external company, and we were not allowed access to the gateway machine. This may seem unexpected at first, but we have found that being denied access to existing system critical resources such as the network infrastructure or the gateway is common practice at most large institutions.

The ELF plugin for Firefox provided an alternative infrastructure to gather HTTP traces. We were given access to two computer labs at the school where we could install Firefox version 3.6.6 and ELF. The first lab was for students and contained 40 machines, 30 of which were in working order. The second lab contained 11 machines, 7 of which were working. We did not log individual user activity because not every user had a unique login or used the same machine on a day to day basis. The school contained another student lab, and several other machines scattered across offices on the campus.

The log format was in simple plaintext that was flushed to disk or uploaded to a central server in a digest every 5 minutes. If the reporting server was unreachable the logs would remain on disk until it could be contacted again. The logs stored on disk were persistent across browser sessions. We elected to upload the logs to our server for convenience, but this could be disabled and the logs could be copied manually if upstream bandwidth were a concern.

Table 2.1: Information and statistics logged by ELF. Each per-event log entry is associated with a timestamp, Window ID, and Event ID.

Information	Logging Rate and Accuracy
IP Address Unique Machine ID Disk Cache Size Clock Skew	Per Digest
HTTP REQUEST / POST - URL - Referer HTTP Header - X-Moz: Prefetch HTTP Header	Per Event
HTTP RESPONSE - URL - HTTP Response Code - Content-Type - Content-Size	Per Event
Page Start Load / Page Stop Load - URL	Per Event
Network Online/Offline	Per Event (5 minute)
User Idle/Back	Per Event (1 minute)

ELF recorded and reported information on a per digest and per event basis. Each digest included aggregate statistics such as cache usage and clock skew. Events such as web requests, responses, cache hits, user idle, and network offline. ELF is configured not to log cookies or any other headers that may be a privacy concern. The complete set of information recorded by ELF is in Table 2.1.

2.2.3 Traffic Analysis

In this section we present some analysis of the traffic traces. We find several well-known results that corroborate with existing web literature, some that corroborate with web in emerging regions literature, and some findings that are completely novel. In this and the next section we compare our results with the two pieces of prior research that are the most closely related to ours by Du et al. [125] and Johnson et al. [157].

Our logs were taken over a 48 day period between September 6 and October 24, 2010. New data continues to be captured for analysis. Our trace contains $1,723,146$ log events with $665,571$ HTTP requests by users, of which $325,037$ were downloaded. There were $207,011$ cache hits, $37,809$ unresponded requests, and $95,714$ (canceled/incomplete/failed). A total of $178,634$ unique URLs were requested from $5,025$ unique domains and subdomains, amounting to over 13.3 GB of data. Note that since we modified the caching behavior of the clients, the caching numbers reflect those changes. Also, prefetched objects are not included in any of the results in this section.

Time and Place of Access

Files downloaded over a period of 2 weeks are shown in Figure 2.1. We see the expected diurnal pattern that is truncated each day due to the hours the school was open. Sundays also exhibit no activity. This pattern was largely consistent except between October 10 and October 17 when nearly no activity occurred due to a holiday.

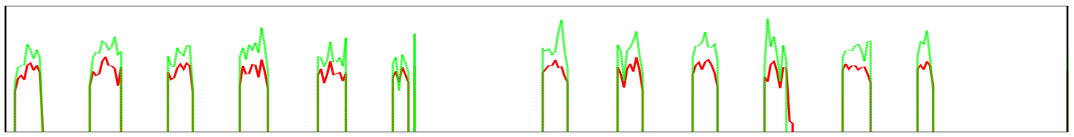


Figure 2.1: Semi-log plot of requests/hour (red) and KBytes/hour (green) over a period of 2 weeks. The expected diurnal pattern is clear for days when school was in session.

The usage level of the machines was split between the two labs. The machines in the teacher's lab ($31 - 36$) viewed approximately 1 to 3 orders of magnitude more pages than those in the student lab ($1 - 30$) in terms of web pages requested. The per byte results are similar. Less than 10% of requests failed or were incomplete; this was slightly lower than the 12.5% observed by Johnson et al. at a rural village in Zambia.

Internet Destinations

The distribution of requests across domains or sub-domains is shown in Table 2.2. Web mail, search portals, and advertising dominate the number of requests. Unlike recent work at rural Internet cafes by Johnson et al., webmail requests dominate rather than Facebook. We assume that this is simply due to the difference in demographic. Unlike the previous result, the demographic of our users is strictly school faculty and students and is unlikely to contain traffic from international visitors. Prior work by Du et al. found more portal requests than mail and advertising requests. In our trace, mail requests were the most frequent followed by portals then advertising.

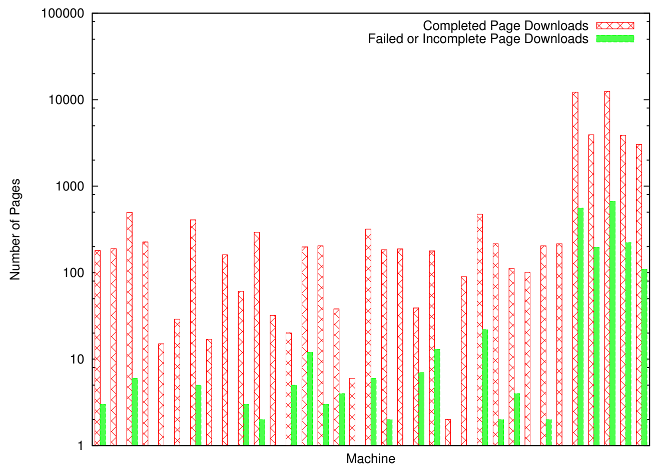


Figure 2.2: Page loads per machine.

Table 2.2: Top requested domains or sub-domains.

Domain or Sub-Domain	Requests
mail.google.com	16.71%
www.google.com	4.35%
*.fbcdn.net	4.52%
*.google.co.in	3.64%
safebrowsing-cache.google.com	1.67%
*.doubleclick.net	2.53%
js.geoads.com	1.43%
ad.yieldmanager.com	1.41%
*.yimg.com	2.16%
www.google-analytics.com	1.00%
other	59.58%

Classification by Type and Size

The MIME types of all files downloaded are tabulated in Table 2.3. The proportion of images to html/plaintext is similar to those in previous studies in rural Internet cafes and kiosks. However, the dominance of video 72.15% is perhaps surprising for an environment without high-bandwidth Internet access. Looking more closely at the video traffic, we found that like in rural Internet cafes, the videos are mostly from google.com or youtube.com. A total of 360 YouTube videos were requested. Of the 360 YouTube videos, 190 were unique and the most popular video was requested 10 times. As with previous results, YouTube video traffic could be dramatically reduced if the videos were cached at the gateway proxy. The actual contents of those videos include educational content, news, TV programming, music videos, and arts and crafts.

Compared to prior work we also found more javascript files downloaded (20% of requests) and nearly no binaries. The amount of shockwave/flash and audio files downloaded was also still relatively low. Finally, the number of MS Office and PDF documents downloaded is negligible.

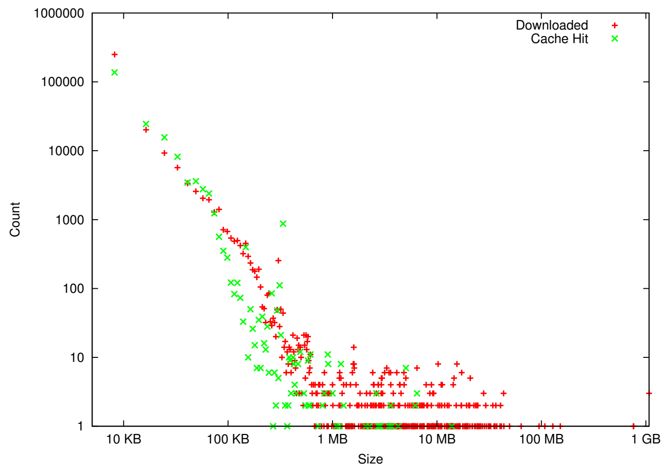


Figure 2.3: Logarithmic plot of file size distribution.

Figure 2.3 shows a logarithmic graph of the size distribution of the URLs downloaded and

Table 2.3: Traffic by MIME Type.

Mime Type	Requests	Bytes
images	**47.46%**	14.11%
html/plaintext	22.30%	1.58%
javascript	20.49%	2.61%
other	5.79%	5.01%
css stylesheet	2.59%	0.43%
shockwave/flash	1.91%	1.83%
video	0.19%	**72.15%**
audio	0.14%	1.20%
msoffice	0.09%	0.31%
pdf	0.04%	0.71%
compressed	$< 0.01\%$	0.06%
binary	$< 0.01\%$	$< 0.01\%$

cache hits at the clients aggregated into 8 KB buckets. We find very similar results to those found by Du et al., with a couple of notable exceptions. We find fewer objects in the midrange around 1 MB, and there are also a few outliers of large files around 1 GB in size that were video downloads. We also observe that the cache hits roughly track the downloaded file patterns but are generally smaller. Also, the largest cache hit is only approximately 14 MB in size.

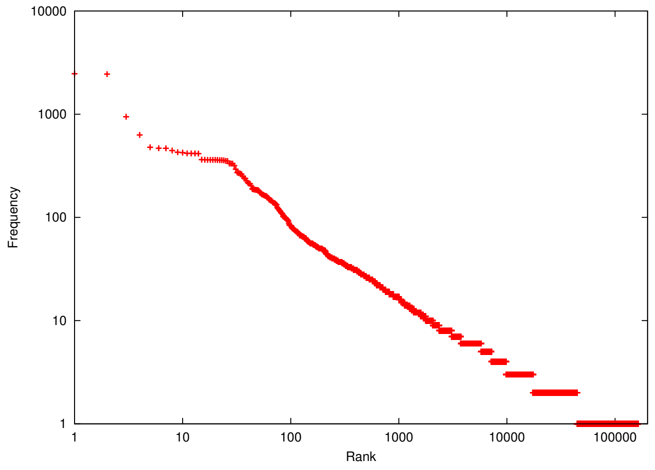


Figure 2.4: Logarithmic plot of URL frequencies by rank. The expected Zipf-like distribution is clear.

Figure 2.4 shows a logarithmic graph of the URL frequencies by rank. The distribution is roughly in line with those in previous studies, with a Zipf-like distribution. We hypothesize that the deviation from the Zipf distribution at the highest ranked URLs $(1 - 100)$ may be due to our smaller population size.

Web Performance

We found that the average time taken for a page to load including rendering time was 3.69 seconds, with a standard deviation of 9.53 seconds. The minimum was 0 seconds, and the maximum was 366 seconds. We also found that the variability was larger across the student machines due to older hardware and presumably malware infections. However, due to the higher number of requests by the teachers' machines, our results are dominated by those faster page load times. This is an interesting result that is not found in most studies due to proxy-based logging mechanisms which cannot capture complete client browser statistics.

2.3 Study 2: Rural University in India

To understand the challenges of web browsing in under-provisioned settings under the conventional browsing model, we observed university students and staff browsing from behind a heavily shared bottleneck connection on their campus in Kerala, India. The network available on the campus was a broadband 8 Mbps connection. However, this single connection was being shared across 400 machines by over 3000 students, staff, and faculty. Furthermore, only 750 Kbps in total bandwidth was allocated to students and staff. A simple estimate of the worst-case average bandwidth available per machine is approximately 1.9 Kbps. This is abysmally slow even compared to dial-up (56.6 Kbps). The average-case bandwidth available across all users was difficult to pinpoint as it fluctuated depending on the number of concurrent users at the time and priority queueing effects at the gateway router. All traffic was routed through a web proxy with a 20 GB disk cache. Our study participants are well educated. While this is not representative of all users in emerging regions, we hypothesize that less educated users may have even fewer coping mechanisms.

2.3.1 Methodology

We used an existing machine in the university's computer lab running Windows XP and Internet Explorer 7. The machine and the network were powered by backup generator to alleviate power outages. The experiments were performed between 12:00pm and 10:30pm. We did not artificially constrain the time of day as the fluctuations in bandwidth were themselves part of the phenomenon of interest.

Fifteen participants (11 male) between 19 and 25 years of age were observed in this study. All participants were enrolled in college, college graduates, or had completed their masters. Five participants were students and ten participants were staff. The participants' fields of study included computer science, electrical engineering, commerce, and business management. Every participant was self-reported to be at least moderately experienced with the web and was able to converse and browse the web in English. The Internet connection that the participants were familiar with varied between dialup and broadband.

Each participant was first given a simple demographic and browsing experience questionnaire. Questions included level of familiarity with the Internet, average Internet usage per day, and comfort while searching and browsing the web. Then each participant was asked to use the Internet for web search using the search engine of their choice to pursue any search topics of their choice. Participants were not informed beforehand as to the exact duration of time available so they would not feel rushed (but were given up to 15 minutes to search). Finally, participants were given a brief semi-structured interview (around 10 minutes) about their experience. Participants were observed and a screen-capture video of their screens was recorded.

2.3.2 Results

The actual bandwidth the participants experienced varied significantly from 20Kbps during peak hours in the early to mid afternoon to 200Kbps in the evening. The reason the bandwidth was higher than the worst-case is likely because not all machines were accessing the network simultaneously. Here we focus our analysis on the participants' web browsing behavior experienced under these conditions.

General Behaviors

Search engine results pages tended to load quickly (under five seconds) due to the mostly-text content and low latency to the search engine servers. None of the users had complaints about the search results page load times during the experiment. The requests for general pages (pages not provided by the search engine) took varying amounts of time to load (ranging from one to 240 seconds) depending on server latency, network congestion, and page contents. The overall statistics for idle time of all page loads in seconds are: mean 12.62, median 4, stddev 26.27. These numbers are inclusive of hits in the local web proxy's cache (10-25% hit rate) and any compression implemented by the accessed web servers. We observed that although many requests were satisfied quickly with 70% pages loading in under 10 seconds, a huge variance the load times of different pages exists with some pages taking up to five minutes to load. We also observed that quickly loading pages were mostly navigational or pages with very little information content.

Participants had no complaints with pages that took less than five seconds to load and were generally only affected if page loading time exceeded ten seconds. When this happened, participants were observed sighing, staring at a blank page loading screen, leaning back, and trying to make conversation with the observer while waiting. This behavior along with self-reported boredom and frustration (e.g. 'I feel angry') generally increased as the page loading time increased. When page load times were over one minute, several participants reported that under normal circumstances they would do something else, give up, wait to browse at a later time, or use a faster connection elsewhere. Four participants reported that they had access to a home broadband connection, and all had access to and were aware of a broadband pay-per-use PC cafe on the campus. As found in previous research [100], participants generally preferred looking only at the first page of results, electing to modify their search query rather than go on to a second page. None of the participants used any advanced search engine options, though most were aware of their existence. Two participants, when asked, revealed that advanced search was not worth the effort to use, preferring instead to iteratively modify their query. Four participants performed image search during the experiment, and two others mentioned they often perform image search.

How Did People Seek to Alleviate Wait Times?

Seven participants (47%) opened multiple windows (the web browser we installed at the school did not have tabbed browsing), and two more reported that they occasionally would. Six reported that outside of the experiment they commonly multi-tasked with other activities including listening to music, using offline applications, reading, or talking to a friend. Some participants who opened multiple windows switched between them while waiting for pages to load.

How Successful were People at Saving Time?

One of the experimenters coded our video data to approximate participants' total idle and busy periods. A participant was defined as busy when he was observed reading the contents of the page or performing any navigational mouse action, and defined as idle otherwise. We found that people who opened multiple windows and switched between them wasted less time (i.e., spent an average of 74% less time being idle). Participants who opened multiple windows, but did not switch between them performed slightly better than those who used a single window (28% less idle time on average). Only when the page loading rate was faster than the user's rate of information consumption (reading speed) was the page load time disguised. We found that some of the improved performance exhibited by the opening multiple windows is an artifact of our particular experimental environment. In our environment, connections are throttled across a shared pool of bandwidth. As a result, when people open multiple windows they effectively increased the amount of bandwidth available to them. What is interesting is that by multi-tasking users self-impose asynchrony to the web search process. Also, even if the bandwidth was not shared across users, people would see gains because data would continue to download on other pages while they view pages already available.

Benefits of Caching and Compression

During our study, the web proxy only had a cache size of 20 GB, and a hit rate of 10-25%. Approximately 27.5% of web servers on the Internet compress files they serve [216]. To estimate the best-case scenario for compression we compressed all files downloaded by our participants during the experiment using a simple compression algorithm (gzip). Unsurprisingly, we found that compressing benefits text files the most (up to 80%) compared to image files that are already compressed, but text files only represent a small (and diminishing) proportion of total webpage sizes on the web [166]. From our idle time measurements and interviews, the existing caching and compression were unable to mask the network latency for our participants. Even with state of the art caching and compression the required orders of magnitude improvement needed to shield the user from delays would not be achieved for intermittent connections.[1]

Observations of Well-known Results

First, most of the searches our participants performed were for textual information, thus most images were not useful despite taking up the majority of download times. This supports previous work that a text-only browsing option could improve satisfaction if images are unwanted [37, 245]. Second, web browsers by default currently only have a progress bar to indicate page loading progress. This feature was not helpful to our participants because it presented only a vague estimate of the time to complete a page load. One participant even claimed to estimate the time to load a page manually based on the rate at which the progress bar filled up. Providing people

[1]Recall that in Chapter 1 we achieved 2.8x overall speedup with our browser optimizations in a less constrained setting.

with a more accurate page load time estimate could allow them to multi-task more efficiently.

We also noticed that our participants often had problems entering effective search queries on their first attempt; their search sessions often required several query reformulations. This affected our participants only slightly, but with high latency or intermittent connections this would be more of an issue. Finally, assistance for search query construction (ie. Google Suggestions) was not always immediately available due to network sluggishness, and when suggestions did appear, they were never explicitly used. When asked why they did not use query suggestions, participants responded that they preferred to iteratively refine their search terms based on previous results.

2.4 Related Work

Web trace data analysis has long history. Our study confirms well-known features of web traffic, including a power law distribution of web requests to their popularity, self-similarity, and a diurnal cycle [96, 97, 250]. Later research on web caching, prefetching, compression, and other acceleration techniques built upon the findings from these traces, highlighting their importance [191, 233].

Early research in user search behavior identified search preferences within the "search and refine" framework [100]. As search systems and user interfaces evolved, studies have tracked user behavior changing along with technology trends [207, 249]. However, relatively little is known about web interaction and behavior in emerging regions particularly among the poorly connected [251].

Studies on web use in emerging regions are typically conducted at a smaller scale and specific to particular contexts. Ratan assesses how Internet-connected PCs affect low-income support staff in modern offices [221]. Closely related work by Du analyzes web traffic in Cambodia and

Ghana at Internet cafes and kiosks [125]. Johnson's work investigates web traffic use in a rural wireless network in Zambia [157]. Other relevant studies have been conducted in telecenters in Mexico [146] and Internet cafes in Kenya and Nigeria [200]. Only recently have attempts been made to characterize web use in emerging regions at a larger scale [148].

Also related to our work is Isaacman's study of web usage in Nicaragua [151]. We target a normal Internet connection where they study web use over a delay tolerant network (DTN) [129]. We do not compare against their work because users behave differently using a DTN; web browsing is converted from a synchronous to an asynchronous activity. This change has a non-trivial impact on browsing patterns as we see in Chapter 5. In addition, some dynamic and interactive content remains difficult to deliver over DTNs that can bias results.

Overall, our studies complement each other and each of these previous efforts to form a more comprehensive understanding of web use in emerging regions.

2.5 Chapter Summary

This chapter studied web browsing in two different contexts in India: the first study was a traffic analysis from a peri-urban primary school near Bangalore, India, and the second study was an observational study of users at a rural university in Kerala, India. The Internet connection at the second study was much more congested than at the first.

From our traffic trace in the first study, we found many similarities to prior work from emerging regions, but also some interesting differences due to either the unique demographic, urban setting, or more recent data. We discovered a dominance of videos (72% by bytes), and significantly more javascript files (20% by requests) than previously measured. We also found that webmail was the most common request followed by portals and then advertisements. We developed a data collection tool for this study named ELF that requires minimal expertise to deploy and use.

In our second study, we find that conventional web optimization mechanisms such as caching and compression are insufficient to provide a positive web experience. The overall approach to web browsing of our participants mirrors results found in previous studies with major differences in behavior apparently caused solely by the poor Internet connection. Participants attempted to multi-task to varying degrees of success proportional to their browsing competence. The penalties of novice web users for mistakes such as typos or poorly chosen links combine to cause a substantially worse experience. Overall, the time wasted by our participants when using the Internet was high and often outweighed the perceived benefits of using the Internet at all.

Together, our results give an indication of where conventional web optimizations begin to fail, and help motivate further research in this area.

Chapter 3

The TCP Breakdown Problem

In extremely congested networks such as those in the previous chapter, we find dramatic fluctuations in page download times. As we show in this chapter, this phenomenon is due to standard congestion control algorithms breaking down. Congestion control algorithms such as TCP-NewReno assume typical per-flow throughput is at least 1 packet per roundtrip time. Environments where this assumption does not hold, a largely unexplored space we call the *sub-packet regime*, are common in emerging regions, where a 128Kbps–2Mbps access link may be shared by 50–200 users. This chapter investigates the impact of pathological-sharing on TCP and other end-to-end congestion control schemes. We find that in the sub-packet regime TCP exhibits severe unfairness, high packet loss rates, and flow silences due to repetitive timeouts. To understand TCP's behavior in this regime, we propose a model particularly tailored to high packet loss-rates and relatively small congestion window sizes. We validate our model under a range of network conditions.

3.1 Motivation

The growth in the popularity of the Internet has contributed to growth in demand for the Internet especially in emerging regions. However, network connectivity has not kept up with demand. As a result, pathologically-shared connectivity is increasingly common in emerging regions. The problem of pathological sharing has been further exacerbated by an increase in the complexity of websites, and an increase in the number of TCP connections created by web browsers. All these factors are contributing to the increased prevalence and increased importance of sub-packet regimes.

In this section, we first define the sub-packet regime. We then motivate the problem by analyzing an end-user's view of web browsing behavior in this regime using real-world access traces. We use simulations of TCP's behavior in the sub-packet regime to explore what makes TCP break down in this regime.

3.1.1 Defining the Sub-packet Regime

We define the *sub-packet regime* as the region of TCP operation when competition between flows results in per-flow fair-shares less than 1 full-sized segment (maximum segment size or MSS) per observed round-trip time (RTT). A TCP flow with segment size S and round-trip time of RTT, is in the sub-packet regime if both of the following conditions hold at the bottleneck link, which has capacity C:

1. number of competing flows, $N \gg 1$, and
2. per-flow fair share is less than S/RTT.

The fair share of a flow on a bottleneck link is inversely proportional to its RTT and is also dependent on the RTT of competing flows on that link. If all flows have the same RTT, then the fair share is C/N. The first condition alone holds in the aggregate links that most Internet traffic goes through, but per-flow fair share on these high-bandwidth links is usually large enough to keep TCP out of the sub-packet regime. The second condition alone holds when one or a few TCP connections use a low-bandwidth link, such as a WiFi or cellular link, but the minimal sharing keeps these links out of the sub-packet regime. It is the *combination* of heavy sharing, on the order of several 10s to hundreds or thousands of competing flows, operating over low-bandwidth networks that causes the sub-packet regime. High levels of sharing is common in emerging regions; we now examine why this sharing is pathological.

We note that statistical multiplexing may cause flows to exhibit sub-packet-regime-like behavior when the average per-flow fair share *approaches* 1 packet per RTT. We thus also consider flows in the region *approaching* the sub-packet regime.

3.1.2 Pathological Sharing: An End-user View

We first examine the web browsing experience at a real university campus in India. The university is equipped with a 8Mbps access link to the Internet throttled down to a pool 750 Kbps for the majority of users. There are approximately 400 computers on campus usable for Internet access. We studied this same setting in study 2 of Chapter 2 and found serious deficiencies in the user experience. Here, we begin exploring the problem by examining object download times recorded by the university's web proxy, constrained to a 2-hour window to minimize the effects of time-of-day load variations. During this period, the proxy recorded 221 unique client IP addresses, and downloaded 1.5GB over the access link. Figure 3.1 shows a scatter plot of download times for objects of various sizes, ignoring cached objects and HTTPS connections, based on logarithmically-sized buckets of object sizes. Figure 3.1 shows the 10th percentile, 90th percentile, min, max, and average download times within each bucket.

The Y-axis spread in the graphs is striking: download times for objects vary by over two

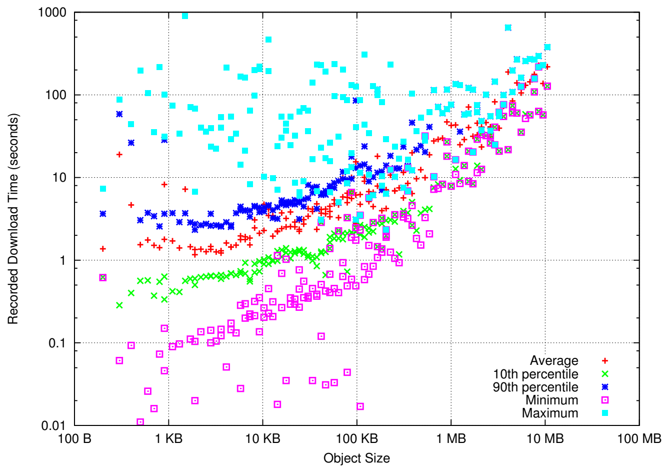


Figure 3.1: Scatter-plot of download times for different object sizes, taken from a 2-hour observation period at the university's Squid proxy. Each raw data point is assigned to a bucket, and the values shown here are the 10th percentile, 90th percentile, min, max, and average values per bucket.

orders of magnitude![1] We can observe that this variation is pervasive: many flows witness long download times across all file sizes. Download time variation eventually decreases at larger object sizes (e.g., 1MB), but most web pages and objects reside in the size region where variation is high.[2]

We make two observations. First, a significant number of users experience poor performance with large download times even for very small object sizes where the entire object can be transmitted in a few packets. Second, the high variation in the download times for comparable object sizes indicates a high level of unfairness across flows. To understand these phenomena, we next use simulations to describe TCP's behavior in the sub-packet regime.

[1] After normalizing server latencies via an emulated dumbbell topology with uniform server latencies we find a reduced, but similar 1.5 to 2 orders of magnitude spread for all but the largest files.

[2] From a crawl of Alexa's top 100 websites (Jan 9, 2010), we found the mean page size (including embedded objects) to be about 350KB and the median to be about 150KB.

3.1.3 TCP in the Sub-packet Regime

The behavior of TCP in the presence of many flows has been studied in prior work [192,212,219]. What is known from these works is that under high contention, TCP flows experience high packet loss rates leading to poor per-flow throughput and unfairness across flows. The small pipe case analysis in Qiu et al. [219] also shows that a small set of flows capture the entire bandwidth while a number of flows remain shut off; however, flows do not exhibit global synchronization problems and link utilization remains consistently high.

Building upon the analysis from prior work, we pinpoint three specific observations about TCP behavior in the sub-packet regime. First, individual flows experience repetitive timeouts frequently that result in long silence periods when flows do not transmit a single packet. Second, flows experiences high levels of unfairness across variable time scales. While long term fairness is better than short term fairness, we observe that flows experience a random selection process where different small sets of flows progress during different short time scales. Finally, none of the existing variants of TCP and TFRC or existing variants of queueing mechanisms (RED, SFQ) address these problems in the sub-packet regime. We now describe these observations in greater detail using simulations. While we have performed several simulations under varied conditions, we present the simplest results that motivate these observations.

Fairness in the Sub-packet Regime

What happens to flow-level *fairness* under pathological sharing? We look at a simple simulation experiment, using a dumbbell topology where flows congest a core link, which uses a simple tail-drop queue. Every edge node (user) spawns two TCP-SACK flows [185]. All traffic is one-way, reflecting download-centric web browsing. Since we wish to focus on congestion control dynamics, which are often obscured by delayed acks, our TCP receivers do not delay acks. The senders use an on-the-wire packet size of 500 bytes.

As the number of flows increases at the congested link, the overall average goodput remains consistently high (greater than 90%), for varying link bandwidths. But fairness among flows suffers, as Figure 3.2 demonstrates using the Jain Fairness Index (JFI) [153]:

$$fairness = \frac{(\sum x_i)^2}{n \sum (x_i)^2} \tag{3.1}$$

The JFI is a value between 1 and $1/n$, with 1 being the best fairness (exactly equal shares), and $1/n$ being the worst (one flow hogs the entire link). JFI values are most readily comparable when the number of flows is the same across the scenarios being compared; otherwise, interpreting the JFI can be problematic because worst-case fairness depends on n. We nevertheless use this index to note deviations from the ideal fairness of 1 since the focus of the experiment is not to measure the exact amount of unfairness in the system.

Long-term fairness is high, as seen for the flows that run for 10000 seconds (17 minutes). Unfairness over shorter periods of 20 seconds sets in, however, as per-flow fair-share drops below 30Kbps, or, with an RTT (including queueing delay) of about 400ms, below 3 packets per RTT. The choice of a 20 second window for our analysis is pragmatic: as expected, fairness becomes worse with shorter windows and better with longer windows.

To better understand how the network manages and how the system distributes bandwidth among sub-packet regime flows, we now take a closer look at the dynamic behavior of the flow aggregate. We simulate a 1Mbps link with a propagation RTT of 200ms and a 200ms packet buffer (1 RTT worth of buffer), leading to a maximum observed RTT of 400ms. Figure 3.3 shows a CDF of flow goodput when 100 flows are sharing this link (a per-flow fair-share of 10kbps or 1 packet-per-RTT). Each curve shows average goodput over a 20-second timeslice.

We make the following observations. First, in any given slice, at most 70 flows are active; the remaining 30 flows in every slice get zero bandwidth. Second, the distribution of goodput

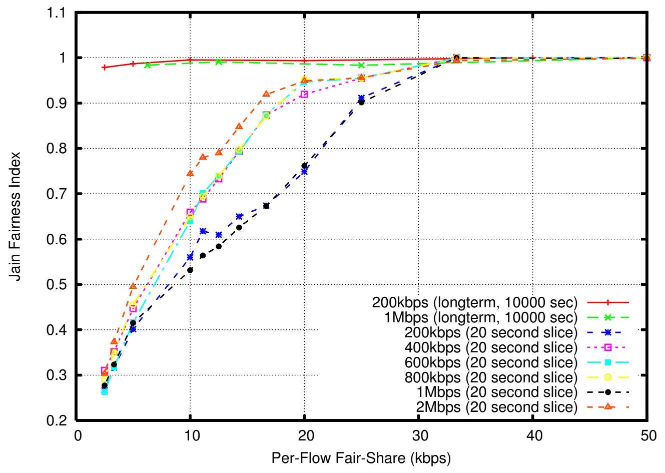


Figure 3.2: Long- and short-term Jain-fairness as a function of ideal per-flow fair-share, for different capacity bottleneck links.

among flows is highly consistent across all the timeslices. While 70% of the flows are active in any given slice, there are 3 distinct groups of flows in each slice: 40% of the flows get about 80% of the link, about 30% of the flows share the remaining 20% of the link, and about 30% of the flows get “locked out” of the system in each slice. Third, no flow starves forever, since, as we’ve seen in Figure 3.2, every flow gets its fair-share of the link in the long term.

These observations lead us to the following significant result: While ensuring that all flows get admitted for equal shares in the long-term, the emergent flow management mechanism in the system is to perform *arbitrary admission control* of flows within shorter time slices. The admission control helps the few admitted flows make progress, but its arbitrariness causes the huge download time variation seen in Figure 3.1.

Repetitive Timeouts

Consider a user who spawns a pool of TCP connections from a web browser. We define a *user-perceived hang* as an event where the user receives no data and observes no progress on the web

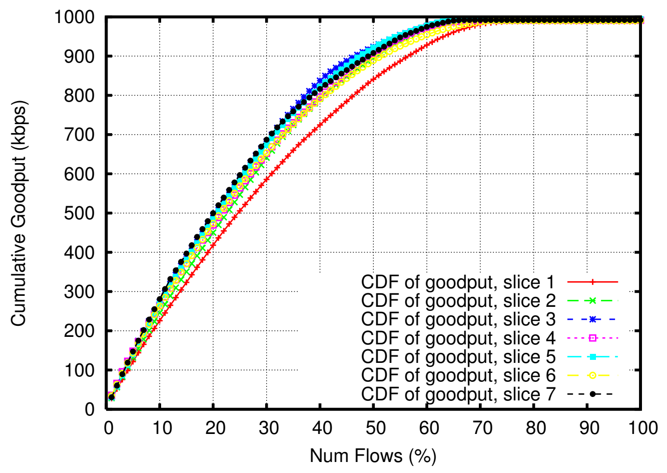


Figure 3.3: CDF, with 100 flows, of TCP goodput in 20-second slices.

browser for some time. The length of a user-perceived hang is the duration during which *none* of the browser's pool of simultaneous TCP connections receives any data. We look at simulation traces of such web transfers in the ns2 simulator on a pathologically-shared link, by creating users that spawn multiple TCP connections each (a web session pool), all sharing a bottleneck link of 1Mbps capacity. The propagation RTT is 200ms, and the bottleneck link has 50 packets worth of buffer space (one RTT worth of delay). The experiment is run for 10 minutes.

Figure 3.4 shows the longest hang experienced by the different users in the system. This figure shows only the single longest hang, not the total of hang times experienced by the user—that number would be higher.

With four flows per user and 200 active users, all users perceive at least one hang time longer than 20 seconds, and with 400 users, almost 50% of the users perceive at least one hang longer than a minute. While spawning fewer connections per user helps reduce congestion, Figure 3.4 also shows that fewer connections worsen the user experience by increasing the chance that all of a user's connections stall at once.[3]

[3]The longest hang time experiences a plateau at 64 seconds that is an artifact of the longest TCP timeout (RTO) period of 64 seconds in the ns2 simulator [211].

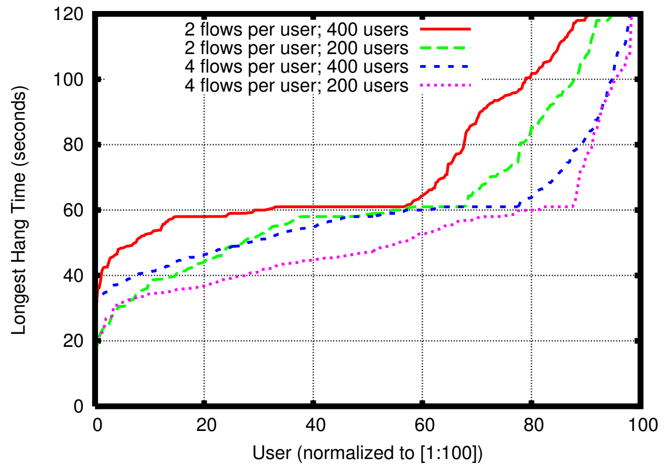


Figure 3.4: Longest *hang* time experienced by users, observed over a 10 minute period. Hang times are sorted in increasing order, and the X-axis shows users normalized to be in the range [1:100].

We briefly describe TCP's behavior under these web transfers. First, the TCP flows experience a high loss rate (30%) that is about the same as the average loss rate at the link. Second, an analysis of the timeout values used by TCP senders for setting their retransmission timers reveals huge variance. Exponentially increasing timer backoffs indicate that flows are repeatedly losing packets, without an opportunity to reset their retransmission timer with a successful new transmission. These repeated timeouts result in increasing silence periods on the link for those flows.

3.1.4 Understanding the Alternatives

We now consider the effect of buffer size on fairness, and briefly discuss what happens with other end-to-end schemes and queueing mechanisms.

Using Larger Buffers

In dealing with pathological sharing, Morris [192] observes that provisioning the bottleneck router to buffer up to 10 packets per flow restores fairness. Increasing buffer sizes simply trades in delay and delay variance for fairness. Figure 3.5 shows the tradeoff required for achieving fairness for a small part of the sub-packet regime. Increasing buffers is infeasible, particularly deep in the sub-packet regime. After increasing buffer sizes, the corresponding increases in delay can be disproportionately high. At 1000 bytes a packet, with 800 TCP connections competing at a bottleneck link of 2Mbps capacity, the maximum queueing delay necessary to achieve fairness is 32 seconds—a delay that most traditional applications are not equipped to deal with, and many others (such as real-time apps) would find unacceptable. Under low load, these large queues result also in large RTT variations that can be problematic for applications and for TCP's RTT estimator.

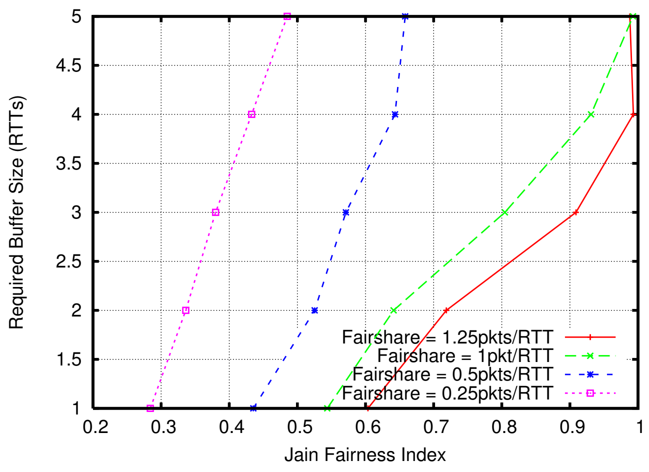


Figure 3.5: Buffer sizes required for restoring fairness, considering 20-second slices.

TFRC vs. TCP in the Sub-packet Regime

TFRC uses equation-based congestion control and pacing, which makes it less bursty than TCP and a distinctly different candidate as compared to other TCP variants. Pacing allows TFRC to perform better than TCP in terms of short-term fairness, as shown in Figure 3.6. but coarse-grained timers in non-realtime operating systems hinder TFRC's ability to pace well [143].

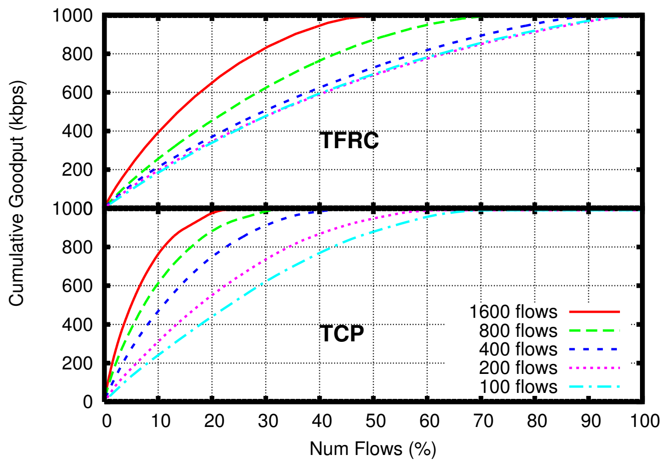


Figure 3.6: TFRC and TCP goodputs over 20-second slices, across different loads.

While TFRC's pacing allows for finer rate control than TCP in sub-packet regimes, TFRC too breaks down, albeit deeper in the sub-packet regime than TCP, when it encounters timeouts.

A noteworthy and unintuitive result is that while link utilization is the same for both, and while TCP is the bursty sender of the two, TFRC observes a loss rate (20.7%) much higher than TCP does (12.8%). Bursty TCP senders send more than one packet in a burst, and a loss episode often results in the entire burst being dropped; such TCP flows appear locked out and are silent afterwards. With TFRC however, the pacing of transmissions results in fewer burst losses within any flow and therefore fewer flows appearing locked out; consequently, while the flows are fairer to each other, more packet losses are observed due to more active senders than with TCP.

AQM and Other TCP Variants

Using other queueing schemes such as Random Early Detection (RED) [135] or Stochastic Fair Queueing (SFQ) [188] under the same settings does not change the behavior of the aggregate by much. RED and SFQ both need much larger buffer sizes for the number of flows being managed, and they fall short of their promise in this regime. XCP [164] offers rate controls, again assuming at least one packet per RTT. Delay-based end-to-end mechanisms such as TCP Vegas [95] break down in this domain as well, since the levels of flow-level contention and queue occupancy are too high, and Vegas too assumes at least a packet per RTT of bandwidth. TCP variants that are in common use—NewReno [134], SACK [185], Compound TCP [238], Cubic TCP [142]—all implicitly assume that the fair share is at least one packet in each RTT. A fair share that is less than 1 packet per RTT sends TCP into timeout periods, since that is the only way TCP can send at at rate lesser than 1 packet per RTT.

3.2 Modeling TCP in Sub-packet Regimes

In this section, we build a simple model particularly tailored for analyzing the behavior of TCP-NewReno in sub-packet regimes with high-packet loss-rates and with relatively small average congestion windows. The main purpose of this model is to analyze the *stationary distribution* of a set of TCP flows, which provides a detailed characterization of the state of a TCP connection. In other words, the model results in the stationary distribution probability, $P(congestion\ window = W)$, for low-values of W, and the probability of a flow being in a timeout or in a repetitive timeout phase. Since the model is ergodic, its stationary distribution corresponds to the probability that a set of flows is in a particular set of states at any point in time; the probability distribution holds irrespective of flow size.

There are several key insights we draw from this model. First, all the transition probabilities in this model are modeled using a single parameter p, the packet-loss probability at the bottleneck link. Second, under conditions where p is roughly a constant, the stationary distribution probability is only dependent on p and is independent of RTT. Third, we observe a shift in the stationary distribution beyond $p = 0.1$ where the probability of repetitive timeouts significantly increases thereby lowering the effective throughput of a TCP flow.

Our model is different from and extends previously proposed models of Padhye *et al.* [201], Fortin-Parisi *et al.* [136]. The fundamental problem with using a generic Markov model for capturing TCP behavior is state space explosion. One of the advantages in the sub-packet regime, however, is that state space is constrained and may be accurately captured using appropriate state transitions. Our model focuses on high loss rates and captures exponentially increasing silence periods due to repetitive timeouts, the dynamics of which are not captured in detail in previous work. The major challenge in the sub-packet regime model that we address is accurately capturing the timeout behavior of TCP flows, with a simple model of small windows. We distinguish our model from related work in more detail in Section 3.5.

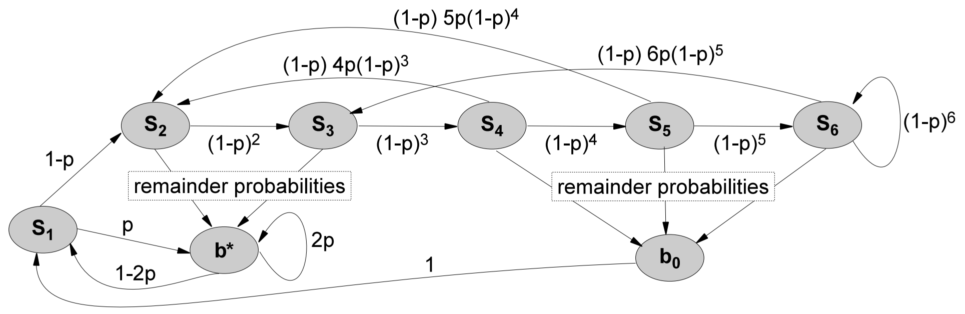


Figure 3.7: The Approximate Model. States S_1 and S_2 get expanded in the full model.

The overall model is described in Figure 3.8. Given the complexity of the model, we will describe how we arrive at this model in stages, by first developing the *approximate model* in Figure 3.7. First, we describe how we start with a simple window-state based Markov model

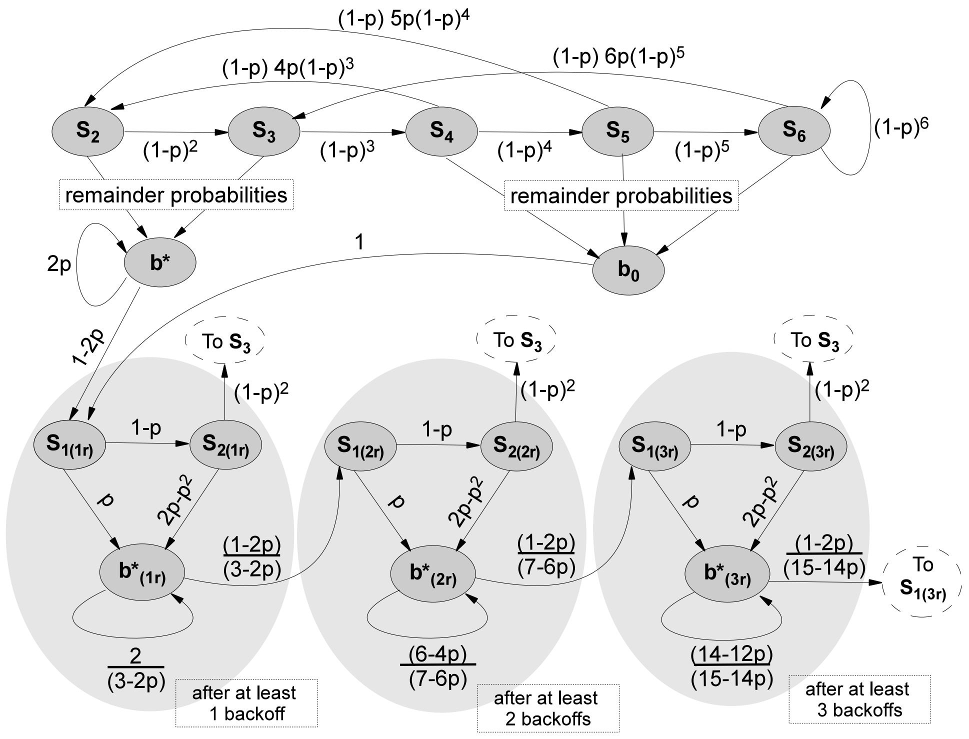


Figure 3.8: The Full Model. This model is limited to a max cwnd of 6, and can be easily extended to larger cwnds.

and how we arrive at transition probabilities. Then, we describe how we model simple timeouts and timeouts with memory of previous backoff value using three aggregate states. Finally, we describe how we expand the model to accurately handle timer backoffs.

3.2.1 Assumptions

We make the following simplifying assumptions. (1) We assume that the TCP flow is operating in sub-packet regimes, with most flows having a small *congestion window (cwnd)* [83] size. We assume a maximum window size in our model, W_{max}; the model may be extended to higher states by increasing W_{max}. (2) We assume that all TCP flows experience medium to high loss-rates in sub-packet regimes. In addition, we assume that the packet-loss probability is modeled using a single parameter p; this is a reasonable assumption since most TCP flows are operating in very low cwnd sizes ($cwnd = 1$ or $cwnd = 2$) resulting in packets of a TCP flow often being single or spaced-out. (3) We assume that traditionally "short-lived" flows do not experience the effects of losses any differently than "long-lived" flows; in sub-packet regimes, even a short flow that carries about 50 packets appears as a "long-lived" flow in the network, particularly due to extended silence periods from timer backoffs. (4) Finally, we assume that the typical *slow start threshold (ssthresh)* [83] is small enough that flows operate in congestion avoidance and not in slow start. With a maximum window size of W_{max}, the maximum ssthresh value is $W_{max}/2$. In this section, we use $W_{max} = 6$ and $ssthresh \leq 3$, but in practice, this could be extended for larger values. $ssthresh \leq 3$ also simplifies the TCP window increment function to be only additive, since window doubling does not go beyond 3.

$$P(S_n \rightarrow S_{n+1}) = (1-p)^n \tag{3.2}$$

The ability to recover from a packet loss without timeout is dependent on fast retransmissions that are triggered by 3 duplicate acks (dupACKs); at least 3 packets must be successfully transmitted in the same window to recover from a packet loss. Therefore, no fast retransmission occurs in our model if a loss occurs when the sender's cwnd is smaller than 4.

While TCP-NewReno is equipped to handle multiple losses in a window [134], studies [230–232] have shown that both TCP-NewReno and TCP-SACK are unable to handle beyond a threshold of losses at low congestion windows. Also, if a flow experiences k packet losses in S_n, to recover using fast retransmit, k loss-free roundtrips are required to recover fully from all the losses [134]. Thus, in sub-packet regimes with high-loss rates, handling multiple losses for low-cwnds is fundamentally hard. In the loss recovery period following the fast retransmit (or, the fast recovery period), dropping any packet will lead the sender into a timeout period. In our model with $W_{max} = 6$, we assume that for states S_4, S_5, S_6, we are able to recover from at most one loss using a fast retransmission, and 2 or more losses result in a timeout at the sender.

The probability that a fast retransmission is triggered in state S_n is given by the probability that exactly one packet is lost, which is $np(1-p)^{n-1}$. The probability that a retransmission, once triggered, is successful is $(1-p)$. Hence, the fast retransmission transition probability from state S_n (for $n = 4, 5, 6$) is given as follows:

$$P(S_n \rightarrow S_{\lfloor n/2 \rfloor}) = np(1-p)^{n-1} \times (1-p) \tag{3.3}$$

In our model for $W_{max} = 6$, there are two circumstances where a sender experiences a timeout: (i) when there are two or more losses in a window, and (ii) when a retransmitted packet is dropped. This probability is simply computed as the residual probability:

$$P(S_n \rightarrow RTO) = 1 - P(S_n \rightarrow S_{n+1}) - P(S_n \rightarrow S_{\lfloor n/2 \rfloor}) \tag{3.4}$$

The upper part of Figure 3.7 shows these transitions put together. Note that S_2 and S_3 do not have fast retransmission transitions, and the sender never reaches a cwnd smaller than 2 through fast retransmissions [83].

3.2.2 Modeling Timeouts

Since timeouts are very common in the sub-packet regime, we capture two forms of timeouts in our model: (a) Simple timeouts; and (b) Timeouts with memory of the previous backoff value (repetitive timeouts). A *simple timeout* occurs when a TCP sender hits a timeout without memory of a previous timeout. In other words, when hitting a timeout, the sender's retransmission timer has a backoff value of 1. On hitting the timeout, the backoff value is doubled to 2. This increased backoff value extends the next timeout period by twice the base timer value, and "collapses" to the base value of 1 only when a new roundtrip time measurement is available, which only happens when cumulative acknowledgements arrive for newly transmitted (not retransmitted) data [211]. A *repetitive timeout* occurs when a flow hits a timeout before memory of previous timeouts is lost. In other words, when hitting a timeout, the sender's retransmission timer has a backoff value greater than 1. On hitting the timeout, the backoff value is doubled again by the TCP sender, causing increasingly extended silence periods.

Modeling Simple Timeouts

In our model with $W_{max} = 6$, we assume that we are in a simple timeout when we transition to a timeout state from a window size of $4, 5, 6$ (states S_4, S_5, S_6) since at least one new packet has been cumulatively acknowledged by the time the flow leaves state S_3 and reaches S_4, thereby resetting the timeout value. For simplicity, we consider the base timeout period $T_0 = 2 \times RTT$. Thus, in the event of a timeout, we capture the $2 \times RTT$ silence period by modeling transitions from S_4, S_5, S_6 to the first timeout state S_1 through an empty buffer state b_0; the transition to state b_0 takes one epoch (RTT), and the transition to S_1 takes another epoch.

Modeling Repetitive Timeouts

When the flow is in small window states of 2 and 3, the timeout value may contain memory of the previous timeouts, which is harder to model due to the Markov chain's memoryless property. As follows, we model these timeouts using an aggregate state to capture the expected wait time before a packet retransmission on a repetitive timeout.

RFC2988 [211] recommends that the maximum timeout period be larger than 60 seconds, and the Linux OS uses a 120 second period. Since these constants are large as compared to typical roundtrip times, we assume infinite timeout states for simplicity of modeling[4]. Let $S_{1/2}$

[4]This assumption leads to our model being limited to $p < 0.5$, as discussed in Section 3.4

represent the state if the sender enters the timeout period with the base timer value of $2 \times RTT$, $S_{1/4}$ if the timer has backed-off to $4 \times RTT$, $S_{1/8}$ if the timer has backed-off to $8 \times RTT$, and so on. These states will get aggregated, and are thus not shown in Figure 3.7.

We model $S_{1/2}$, $S_{1/4}$, $S_{1/8}$, and other other infinite timeout states as follows. If a sender enters at $S_{1/4}$, it must wait for 3 idle periods before retransmitting, and if a sender enters $S_{1/8}$, it must wait for 7 idle periods before retransmitting. On a successful retransmission, which happens with probability $(1-p)$, the sender leaves the timeout state and enters S_2, since the new cumulative acknowledgment received in response to the retransmission increases the sender's congestion window to 2. On an unsuccessful retransmission, which happens with a probability p, the sender doubles its timer period and enters the next longer timeout state.

To be able to incorporate these infinite timeout states into our Markov chain, we aggregate them into two states: a buffer state (b^*) where the sender waits for an amount of time, and the retransmit state introduced earlier (S_1), where the timer goes off and the sender retransmits. With these aggregated states, the following transitions are possible:

- From a small congestion window (S_2, S_3) entering a timeout period, $S_n \rightarrow b^*$;
- staying idle, $b^* \rightarrow b^*$; and
- transition from idle period into retransmission state, $b^* \rightarrow S_1$.

The expected wait time at the aggregated buffer state, b^*, may be calculated as follows. A TCP flow waits for 1 epoch in $S_{1/2}$ before entering the retransmit state, 3 epochs in $S_{1/4}$, 7 epochs in $S_{1/8}$, and so on. Thus, once in a timeout period, the expected amount of time that a TCP flow must wait in the timeout period before retransmitting is computed as:

$$\begin{aligned} Expected\ idle\ time = & P(S_{1/2} \mid RTO) + 3P(S_{1/4} \mid RTO) + \\ & 7P(S_{1/8} \mid RTO) + ... \end{aligned} \tag{3.5}$$

To resolve equation (3.5), we note that:

$$P(S_{1/2n} \mid RTO) \div P(S_{1/n} \mid RTO) = p \tag{3.6}$$

We also know that

$$\begin{aligned} P(S_{1/2} \mid RTO) + P(S_{1/4} \mid RTO) + \\ P(S_{1/8} \mid RTO) + \ldots = 1 \end{aligned} \tag{3.7}$$

Combining equations (3.6) and (3.7), we get

$$\begin{aligned} &P(S_{1/2} \mid RTO)(1 + p + p^2 + p^3 + \ldots) = 1 \\ &P(S_{1/2} \mid RTO) = (1 - p) \end{aligned} \tag{3.8}$$

Equation (3.5) may now be resolved using equations (3.6) and (3.7) to give the expected idle time in the timeout state as follows:

$$\begin{aligned} Expected\ idle\ time &= 1(1-p) + 3p(1-p) + 7p^2(1-p) + \\ &\quad 15p^3(1-p) + \ldots \\ &= 1/(1-2p) \end{aligned} \tag{3.9}$$

Thus,

$$\begin{aligned} P(b^* \to S_1) &= 1/(Expected\ idle\ time\ in\ b^*) \\ &= 1 - 2p \end{aligned} \tag{3.10}$$

Consequently, the probability of staying idle,

$$P(b^* \to b^*) = 2p \tag{3.11}$$

Finally, on a successful retransmission in S_1, the sender enters S_2 with a probability of $(1-p)$, while an unsuccessful retransmission leads the flow back into the timeout state, b^*, with a probability of p.

Capturing Timer Backoffs Accurately

To model repetitive timeouts more precisely, we need to break up the timeout retransmit state S_1 further as illustrated in the bottom part of the expanded full model in Figure 3.8. So far we have assumed that after one successful timeout retransmission, the flow re-enters the state S_2. What this transition misses is that while the sender's cwnd grows to 2 on successfully retransmitting after a timeout, the sender's backoff value has not yet collapsed, since TCP needs an ack for a *new* data transmission (not a retransmission) to be received to collapse the backoff value.

The approximation used so far of going back to b^* from S_1 on a timeout while simple, loses information. We know that the sender has timed-out at least once when the flow is in state S_1, so we may put a larger lower-bound on the amount of time that the sender will be in an idle state on a subsequent loss.

After one timeout, we arrive at S_1 (either with memory or without) and the minimum timeout value is $2 \times RTT$, resulting in an expected wait time of $1/(1-2p)$. However, if we experience a loss at S_1, the minimum timeout value is $4 \times RTT$ with a minimum wait time of $3 \times RTT$. This may be viewed as the same infinite state machine described earlier in equation 3.5 to represent infinite timeout states, but starting here with 3 wait states at $S_{1/4}$. The aggregate representation of this new infinite state may be modeled using a single state with a corresponding expected wait time calculated as:

$$\begin{aligned} Expected\ idle\ time &= 3p(1-p) + 7p^2(1-p) + \\ &\quad 15p^3(1-p) + \ldots \\ &= (3-2p)/(1-2p) \end{aligned} \tag{3.12}$$

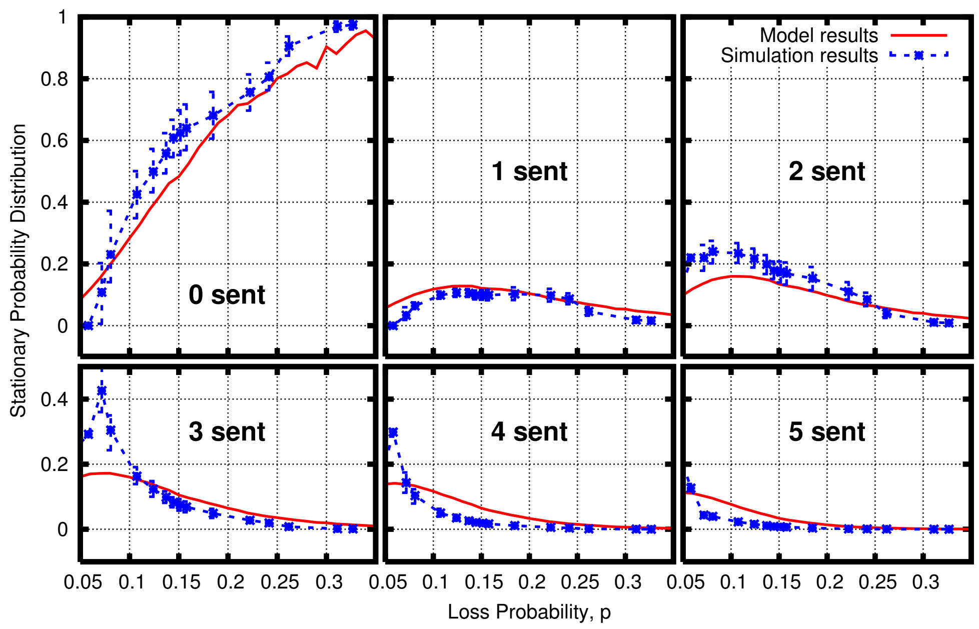


Figure 3.9: Stationary probabilities from the model and from simulations for 6 of the sending states that a TCP connection can be in. Error bars show 10th and 90th percentile flow values.

This may be represented by breaking up the S_1 and S_2 states in Figure 3.7 using an aggregated three state transition diagram of $S_{1(1r)}, S_{2(1r)}$ and $b^*_{(1r)}$, as shown in the bottom left part of Figure 3.8 (grayed oval). $S_{1(1r)}$ represents the retransmit state after the first timeout, $S_{2(1r)}$ represents the state where a new transmission is attempted after a successful retransmission in $S_{1(1r)}$, and $b^*_{(1r)}$ represents the aggregate buffer state for idle time on the first repetitive timeout (or, a timeout after at least one backoff), which occurs if the retransmission in $S_{1(1r)}$ is unsuccessful. Note that $S_{2(1r)}$ now becomes a distinct state, separate from S_2. On successful transmissions of the entire window in $S_{2(1r)}$, which happens with probability $(1-p)^2$, the flow loses backoff memory and enters state S_3, since we expect *new* transmissions to be sent in $S_{2(1r)}$.

To model just one repetitive timeout, we could put a loop from $b^*_{(1r)}$ to $S_{1(1r)}$. However, to model further repetitive timeouts for better accuracy, we replicate the $3-state$ diagram to $S_{1(2r)}, S_{2(2r)}$ and $b^*_{(2r)}$—states representing a second repetitive timeout (or, a timeout after at least two backoffs)—where the minimum backoff is $8 \times RTT$. The expected wait time is calculated as $(7-6p)/(1-2p)$. To model a third repetitive timeout, we add three more states with a minimum timeout of $16 \times RTT$. This is modeled in Figure 3.8 with the states: $S_{1(3r)}, S_{2(3r)}$ and $b^*_{(3r)}$.

While more repetitive timeout states may be similarly modeled, we end our model in Figure 3.8 at three repetitive timeouts, with a loop back from $b^*_{(3r)}$ to $S_{1(3r)}$.

3.3 Validating the Model

We validate the model using ns2 simulations of TCP flows operating in sub-packet regimes. We use a simple network topology with a single bottleneck link. TCP-SACK is used at the endpoints and the simulations are run for 5000 seconds each. We measure loss rate as the fraction of packets that are dropped at the bottleneck queue and consider epochs of average-RTT size to estimate the senders' probabilities of being in one of the model's states. We get the different loss rates shown in the figure by modifying the number of competing flows at the bottleneck link.

Figure 3.9 shows the model's predicted probability distribution for varying loss rate p, overlaid with results from simulations where we measure and plot probability against observed loss rate. For this simulation set, the flows all have a propagation RTT of 200ms, the bottleneck capacity is 1Mbps, and the bottleneck link is equipped with an RTT's worth of buffer (50 packets, at 500 bytes per packet).

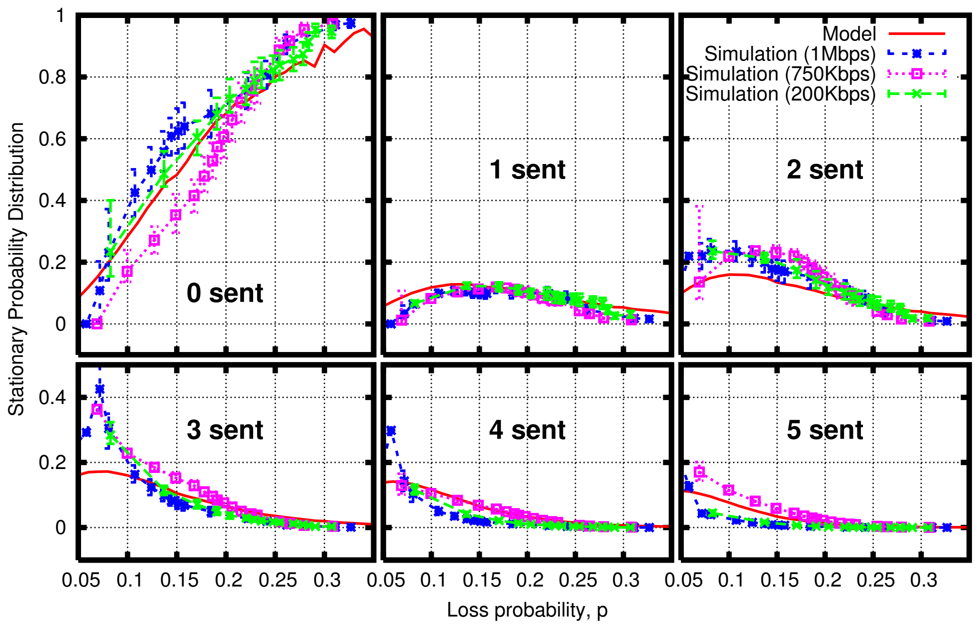


Figure 3.10: Validating the model with different bottleneck bandwidths.

Note that "0 sent" is the sum of probabilities for all the b* states in the model, and similarly "1 sent" and "2 sent" represent the sums of the S_1 states and S_2 states, respectively. The other graphs are self-explanatory[5]. Simulation results agree well with our model at loss rates greater than or equal to $p = 0.1$. We note that the simulations slightly differ from our model for $p < 0.1$ for the following reason: under lower loss rates flows grow to window-sizes larger than 6, whereas state S_6 in our model (Figure 3.8), representing a flow window-size of 6, is a "limiting" state. Flow window-sizes that are larger than 6 are considered anomalies that the model does not capture. There will thus be some probability that flows back from S_6 return to the lower states in our model that will not occur in reality when loss rates are low.

Figure 3.10 shows results for simulations under a variety of link bandwidths (up to 1Mbps),

[5]We do not show S_6 in this graph, but the agreement is similar.

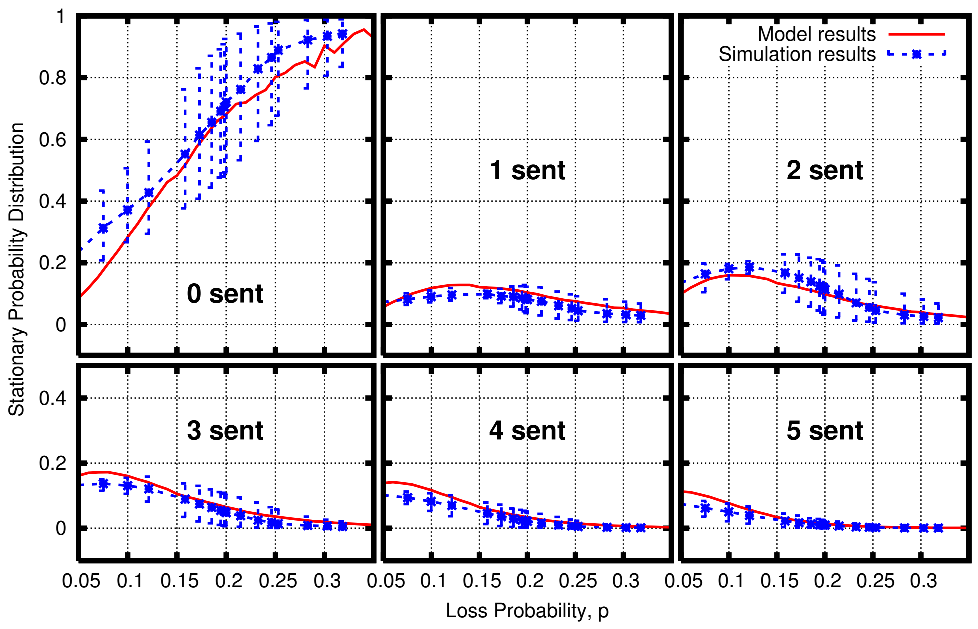


Figure 3.11: Validating the model with mixed-RTT flows.

The buffer size for each simulation was set to an RTT's worth. The simulations all agree equally well with the model. Figure 3.11 shows results from a simulation set with mixed-RTT flows. We simply divide the flows evenly into eight groups, and each group is assigned one of eight RTTs chosen randomly between 200ms and 400ms. While measuring the average probabilities, note that since the flows have different RTTs, epoch-sizes used for the different flows are different, and depend on the flow's RTT. We find that simulation results agree well with our model.

We also ran simulations under a variety of propagation RTTs, and under RED and SFQ AQM schemes, and obtained similar agreement with the model. Note that RED and SFQ do not significantly influence the behavior of TCP aggregates in the sub-packet regime, and thus do not impact the agreement of simulation results with the model.

3.4 Model Strengths and Limitations

The most important strength of the model is that it provides detailed information about the states of a TCP flow in the sub-packet regime. This information is more fine-grained than the expected throughput (as predicted by the Padhye model [201]) and can be used to control flow dynamics in the network. Our model predicts the stationary probability distribution of a flow across its different states, which is more detailed information and therefore harder to estimate than an expected value of a single flow parameter such as throughput. The stationary distribution of a flow across states is representative of the long-term behavior of a flow; hence, a single trial run of a flow may not exactly follow the exact distribution unless the flow is run over a very long time period. This explains the variance in the actual observed values in the validation experiments.

Our model is specifically designed to accurately capture the details of the TCP's timeout behavior including repetitive timeouts, idle time, and backoff behavior. While the example version of the model presented in this Section is for $P_{max} = 6$, we can certainly extend the model by adding states higher than S_6, to more accurately capture TCP behavior with larger windows. However, our model is not designed for analyzing TCP behavior under low-loss conditions ($p < 0.05$) where timeouts are a relatively rare occurrence. The model can also be extended to handle other variants and extensions of TCP such as Early Retransmit [82] (that modify the loss detection and recovery behavior of TCP) by simply adjusting the probabilities of the fast-retransmit transitions in the model.

Our model trades-off broader applicability for simplicity. Many of the states in our model, such as the wait states, are aggregate states that model the expected behavior across sets of infinite states. This aggregation achieves two important simplicity goals: it avoids state space explosion, and it allows for a compact representation of the full state diagram. The model still achieves a good approximation of the stationary probability distribution of a flow at a given loss rate albeit only within the sub-packet regime. Our model is also simple in that it depends on only parameter p, the loss rate. A TCP flow that sends several packets in a short time period may experience bursty loss patterns; however, a TCP flow in a sub-packet regime transmits very few packets every RTT and packet losses for such flows are more spread out. Hence, using a single loss parameter p is a reasonable approximation for flows within this regime. The state transitions in the model are at the granularity of RTT epochs, which was done to make the model independent of flow RTT. As shown in the validation, the model is fairly accurate even when flows have different RTTs and when the bottleneck capacity varies.

Our model is not usable for $p > 0.5$ due to the self-loop at state $b*$ in Figure 3.8. While this may appear restrictive, it actually identifies a potential instability in TCP. Our model uses the assumption of infinite backoffs with TCP's retransmission timer; without a limit on the backoffs, TCP becomes unstable for $p > 0.5$, with expected timeout idle periods of $\inf$. It is possible to replace the infinite timeout assumption with a finite timeout to extend the model for $p \geq 0.5$. While this may be of theoretical interest, experience indicates that a loss rate of 50% is not actually sustainable even in the context of pathological-sharing.

3.5 Related Work

The past three decades have seen a tremendous amount of work on analysis of TCP congestion control. We outline only work closely related to ours.

The earliest work on pathological-sharing among TCP flows by Morris appears about a decade ago [192]. Morris recognizes the limitations of TCP operating in regimes where the fair share is under a single packet per roundtrip time and provides some insight into observed flow behavior and aggregate behavior at the bottleneck. Our work is an extension of this previous line of analysis. As discussed in Section 3.1.4 we find Morris' solution to be impractical; our solution is more practical and deployable.

TCP's loss of fairness under pathological sharing is not a surprise, and has been noted in [184, 212, 219]. We build on these previous observations, and contribute a more systematic understanding of the dynamics that dominate in these regimes. Even in simply observing fairness results from simulations, we find that the emergent admission control behavior in the short- to medium- term, and long-term fairness are novel observations that inform our understanding of the problem.

Prior work in stochastic models for TCP, such as [80, 85, 102, 171, 173, 186, 190, 201, 235], derive analytical expressions for the steady-state throughput of a TCP flow based on its roundtrip time and loss probability. Of these models, ours is closer to [102,201], which consider a discrete-time model and a discrete evolution of the window size. The Padhye model is a much better fit when the packet loss rates, p, are relatively small; at high values of p, however, we observe extended and repetitive timeouts, the dynamics of which are not captured in detail in the Padhye model. While the Padhye model provides the expected average throughput, our stationary distribution is a more complete characterization of the state of a TCP connection. Finally, one subtle difference is that, the value of p in Padhye model represents the probability of loss indication (or loss episodes) while we explicitly model the (in)ability of TCP to recover in the face of a bursty loss of several packets.

Our TCP model supplements Markov models for TCP that have been proposed in [136,139]. These models focus on TCP behavior when the packet loss rates are relatively small. Specifically, Fortin-Parisi-Sericola (F-PS) [136] build an extensive model that is built to yield expected goodput. Our model yields a detailed characterization of the states of a TCP connection, which is fundamentally harder than finding the expected goodput. Yet, and despite the complexity due to modeling repetitive timeouts, our model is simpler and more intuitive than the F-PS model because we assume the sub-packet regime, high loss-rates, and small windows.

Work in low bandwidth access links typically assumes a low degree of sharing at the link. For instance, Spring *et al.* [240] and Andrews *et al.* [87] both propose solutions for improving performance at low bandwidth access links, but both operate under low degrees of sharing and assume per-flow fair share of at least 1 packet per RTT.

3.6 Chapter Summary

In this chapter, we showed how TCP breaks down in the sub-packet regime. As TCP probes for bandwidth, it cannot send fewer than 1 packet per RTT in a controlled manner. Consequently, in the sub-packet regime, TCP uses the only coarse-grained mechanism it has available to send at a lower rate than 1 packet per RTT—timeouts—yielding the individual and aggregate effects that we detailed.

To the best of our knowledge, our work is the first to formally characterize the sub-packet regime and highlight its importance. We characterize TCP behavior in this regime and show that TCP and its many variants under different queueing conditions result in severe unfairness, high packet loss rates, and flow silences due to repetitive timeouts. We have proposed a detailed model to characterize the TCP dynamics that dominate in this regime. We briefly validate this model under a range of network conditions.

Chapter 4

Fixing the TCP Breakdown Problem: A Middle-box Approach

The TCP model we presented in the previous chapter may be applied in a variety of ways at a middlebox to both predict the status and behavior of a flow as well as to potentially design non-intrusive middle-box solutions to enhance performance in sub-packet regimes. In this chapter, we outline some potential applications of the model. We then describe one instantiation of our ideas that fixes the TCP breakdown problem using a middle-box approach. We show that through a combination of admission control and fine-grained packet drop prioritization TCP fairness may be preserved, flows once again behave predictably, and average object download times are dramatically reduced.

4.1 Applications of the TCP Model

In this section we briefly describe several ways that the model and insights from the previous chapter may be applied.

4.1.1 Predicting Network and Flow State

The most straightforward application of the model is to predict the state of the flows across a bottleneck link using a middlebox. Given the aggregate loss rate at the bottleneck, the model currently gives us the probability of finding a flow in one of several states. Similarly, the distribution can also be used to estimate the fraction of flows that are currently in timeout states on a pathologically-shared link. While these aggregate measures are useful for monitoring purposes, a key value of the model is in its ability to predict the possibility of a timeout or a repeated timeout. For a single flow, the model can be used as a mechanism for estimating the probability of hitting a timeout state. Specifically, given (i) the aggregate loss rate at the bottleneck, (ii) the current state of a flow, and (iii) the size of an epoch (flow RTT), the middlebox can track the number of packets within each epoch and predict the probability that a flow could potentially hit a timeout state in the following epoch. All of this information is available to a middlebox at the pathologically-shared bottleneck link. Passive RTT estimation techniques can be used to estimate epoch size for a flow, and aggregate loss probability can be observed. For a middlebox managing the pathologically-shared link, this information is particularly useful in advance to implement a range of different flow-state aware queue management policies. We outline some of the possible queue management policies next.

4.1.2 Middlebox Queue Management

A middlebox could leverage the model to control the behavior of a flow by altering its packet drop policies on a per-flow basis. For example, a middlebox can use the probability of a packet drop triggering a flow timeout to dictate the priority of a packet drop (or transmission) in several ways; we discuss two ways below.

A simple scheduling policy at a middlebox can use different service classes for flows based on the probability of a flow hitting a timeout or a repetitive timeout. More vulnerable flows are placed in higher service classes and less vulnerable flows are placed in lower service classes; the scheduling queue management policy at the middlebox can be careful to drop packets from lower service class flows before going up to higher service classes.

Our model uses number of losses in an epoch for a flow to predict its next state; a middlebox could similarly control the state changes of a flow by controlling losses. In this case, the middlebox performs per-flow state monitoring and keeps track of how many packet drops a flow can sustain without a timeout, and uses this information to make a decision about an impending packet drop at the bottleneck queue.

While we can thus achieve finer control over packet and drop priorities using our model at a middlebox, we now show, to a first degree of approximation, how such control can be effective by outlining a simple optimization that prioritizes retransmissions.

Repetitive timeouts are a more serious problem in sub-packet regimes and they are triggered due to the loss of retransmitted packets. While packet losses trigger a reduction in window sizes, loss of retransmitted packets incur the high cost of increasing timeout periods, resulting in some flows appearing to be “shut-off” for a while, until the retransmission(s) followed by at least one new transmission all get through. It is difficult for an endpoint in the flow to control whether a retransmission gets dropped in the network, but it is possible for an intelligent router/middlebox at the congested link to distinguish and prioritize scheduling of retransmissions. In sub-packet regimes, the ISP providing a pathologically-shared access link could implement and deploy such a router/middlebox.

4.1.3 Admission Control

Under scenarios of extremely heavy pathological sharing where the fair share of a flow is less than 1 packet per RTT, timeouts for a certain fraction of flows are unavoidable. One important takeaway from the model, is that beyond a loss rate of 10%, the stationary probability of flows being in a timeout state significantly increases. Under such conditions, a middlebox could choose between one of two different policies: (a) use a scheduling policy that provides fairness across all flows where timeouts are distributed across time equally across flows; (b) use explicit admission control policies to shut off a few flows and operate the bottleneck link in a safe regime where the aggregate loss rate for admitted flows is less than 10%. Using admission control policies, a middlebox can ensure that at least a few flows make progress.

4.1.4 End-host Optimizations

A subtle observation from the model is that there are cases where the backoff value increases even after some retransmissions actually get through and are acknowledged. We find that often a retransmission may get through after a timeout period, but the subsequent new transmission(s) are dropped. Since an acknowledgment for a retransmission is simply not used by TCP to update the RTT estimate, dropped new packets are often recovered through an extended timeout. Being able to update the timers on receiving acks for retransmissions is likely to be a useful performance optimization in sub-packet regimes. This could be accomplished at the sender by disambiguating between ACKs for transmissions and acks for retransmissions using the Eifel Algorithm [181].

4.2 A Middle-box Approach

The primary design goal of our approach is to prevent TCP breakdown in low bandwidth network conditions especially when a majority of flows are forced to operate in the sub-packet regime.

Our goal is to determine a *non-intrusive* in-network solution that can improve performance, fairness and predictability without making any modifications to the end-hosts and the application layer. Two aspects are noteworthy from our previous analysis in Chapter 3: (a) at high contention levels, a random fraction of the flows are shut out for short time periods yielding arbitrary flow admission control; (b) loss of retransmissions or new transmissions immediately following re-transmissions are particularly expensive as they lead to backoffs and extended silence periods. Both observations indicate situations that an ideal solution would like to avoid, and we use them to motivate our approach.

Our solution uses the lessons learned, and represents a first-order approximation of one possible ideal solution. Our solution approach leverages a combination of two in-network middle-boxes that are positioned at either end of the low-bandwidth link in both the forward and reverse path of a TCP connection. In practice, our solution can be implemented at a router level or can be realized using a combination of software routers or transparent proxies (which tunnel the traffic between them) at either end of a low bandwidth link.

For simplicity of discussion, we assume that we have control over the low-bandwidth network resources and the underlying channel has very low loss rates and all losses are caused due to congestion at the middle-boxes. In reality, if the middle-boxes are overlay nodes where the traffic between them experience unpredictable losses due to cross traffic, then we would build our entire solution on top a system such as OverQoS [242], which provides a controlled-loss virtual link abstraction with very low loss rates between the two overlay nodes. Unless we have control over which packets are dropped at the middle-boxes, it becomes fundamentally hard to provide any form of quality of service in these highly constrained sub-packet regimes.

Our solution uses a combination of several known techniques. First, we identify *flow pools* of inter-related flows corresponding to the same application. Second, we perform *admission control* of flow pools based on the current congestion level in the network. Given the scarcity of network resources, admission control is essential to prevent congestion collapse of flows and extended timeouts. Third, we provide *fair sharing* and isolation across flow pools. Fourth, we perform *fine-grained packet control* on a per-flow basis where we give higher priority for *packet retransmissions* to reduce the probability of timeouts of admitted flows. We next outline these steps in greater detail.

4.2.1 Flow Pools and Admission Control

To arrive at an appropriate admission control policy, we first need to understand common application traffic characteristics. Many applications setup separate streams; the canonical example being HTTP web traffic. In the case of $HTTP/1.0$ a separate TCP connection is setup for each request, and in $HTTP/1.1$ requests may be pipelined.[1] A typical web session may simultaneously retrieve content from several web sites in parallel. We simply define a "flow pool" as a collection of inter-related flows from the same source to different destinations that are all initiated within a short time-period (typically set to a few seconds).

A flow is admitted if: (a) it belongs to a flow pool that has already been admitted, or (b) it belongs to a new flow pool and the current number of flow pools admitted is below the maximum number of allowed flow pools $Flows_{max}$. The $Flows_{max}$ is determined empirically based on

[1]While this may be true for $HTTP/1.1$, in practice, it not supported in a large proportion of web servers. Additionally, many of the newer browsers such as Firefox 3, and Internet Explorer 8 aggressively create new TCP connections to servers (up to 8 persistent connections per server by default) in a counter-productive (for this regime) attempt to download more quickly.

the number of flow pools that can be accommodated by a given network while maintaining the loss-rate below a fixed threshold $p_{thresh} = 0.1$ derived both empirically and from a model we developed previously in Chapter 3.

When a user initiates a new flow, the middle-box checks if the flow belongs to an existing admitted flow pool and if so, the connection is allowed. Otherwise, the admission controller is invoked to verify if a new flow pool can be initiated. If a flow is not admitted, then the SYN packet is dropped. In certain applications such as web browsing, one can make simple modifications to provide feedback to the user on the expected wait time or a spoofed HTTP 503 "Service Unavailable". If the middle-box acts as a proxy (instead of a transparent proxy), it can provide useful feedback as an appropriate response as maintaining a visible queue of requests with expected wait times and finish times for each browsing request. Such an approach has been integrated in prior work on delay-tolerant web browsing solutions [245].

In a feedback oriented solution, after a specific wait time, T_{wait}, the user is guaranteed admission for one flow pool. The admission controller will admit a flow pool after T_{wait} regardless of the number of already admitted flows. Therefore an accurate estimate of T_{wait} is required to keep the number of admitted flows near the desired operating point. If the T_{wait} is not communicated to the user, then we allow the first SYN packet from the user after the T_{wait} if the TCP connection is still alive in timeout phase.

4.2.2 Fair Scheduling and Tracking Progress

At the flow pool level, we employ a fair share based scheduling policy where each admitted flow pool is allowed a fair share, F_s of the total bandwidth to allow all flows to make predictable progress. F_S is independent of the number of connections within a flow pool. Hence, if a user spawns several connections in a flow pool, these connections compete within themselves for the share F_S. A flow pool exceeding its fair share experiences packet drops unless additional capacity is available due to inactive flow pools. Inactive admitted flow pools are removed after an idle period exceeding a timeout. During inactive periods, the fair share of a flow can be used by other competing flow pools. Every flow pool can accumulate *debt*, D depending on the actual usage levels. To be fair across users, the next time a flow pool attempts to enter the system, the admission controller uses the debt as a measure to compute an additional penalty wait time for re-admitting the flow pool. While a flow pool is inactive, D decays over time at a rate F_S to zero.

4.2.3 Fine-grained Packet Drop Prioritization

After a flow pool is admitted, the middle-box tracks the status of each individual flow within a flow pool. Here, our goal is to keep individual flows in active state without hitting timeouts. Thus, over fixed epoch time-periods we maintain a simple counter $N(f_i, k)$ of the number of packets transmitted by flow f_i in epoch k. We try to set the epoch time-period based on the average RTT observed by flows but this estimate may not be accurate. We use $N(f_i, k)$ to track the activity of a flow and determine the drop probability of the flow f_i.

Within a flow pool, we use three basic steps in our packet drop and packet prioritization solution. First, we give higher priorities to packet retransmissions to reduce the possibility of timeouts. Second, we use the status of the flow to determine the packet drop priority of a flow. Finally, we make no specific distinction between different flows within a flow pool unless provided with special application-level knowledge to distinguish between flows in a flow-pool. I.e. For web browsing traffic the destination IP addresses and the order of connection arrivals could be used to derive a relative importance across flows in a pool.

When possible, retransmissions are accorded the highest priority and are not dropped. In addition, a priority scheduler is implemented to keep most flows in a "good" operating range. For each flow f_i, we track $N(f_i, k)$ and if $N(f_i, k)$ is very small (≤ 3), we accord a very low drop priority for f_i. For higher values, $(N(f_i, k) > 3$, the drop priority of a flow f_i is directly proportional to $N(f_i, k)$. Once a flow experiences a drop, its drop priority is halved signaling a window halving by TCP.

4.3 Implementation

We implemented our middle-box solution using the Click Modular Router [168]. In our implementation, separate Click elements are responsible for admission control, fine-grained packet scheduling and for prioritizing retransmitted packets. The existing Click elements provided most of the core functionality to implement our in-network solution. The remaining functionality was implemented in 1500 lines of C++.

4.4 Evaluation

In this section, we evaluate the effectiveness of our middle-box approach to fixing the TCP breakdown problem with emulation-based tests and real-world web traces from our prior deployments. In our evaluation, we show that our solution is able to solve the TCP breakdown problem and is able to achieve fairness, better object download rates and predictability. To evaluate our solution we use a controlled setup where our test bed network is configured with multiple machines on a LAN over a 10 Gbps switch. We use the Click modular router to emulate a bottleneck link for a specified bandwidth, delay and buffer settings. We use the web browsing traces from our deployments to emulate the browsing behavior of multiple users who compete over the low bandwidth network. Each client is provided a virtual IP address and is given a log file that corresponds to the browsing pattern of real user. Each client reads the log file, normalizes the time-stamps to the current time, and replays each request at the appropriate time. In all configurations the bottleneck link is set to 1 Mbps, 200ms RTT, with a total queue size of 25 packets (bandwidth * delay * 1000 byte packets).

Fairness

To evaluate short term fairness, we first repeated the experiment from our simulation of running several long term flows through our emulated bottleneck link at once. For our system this equates to a single flow per flow pool, and admission control does not actually come into play. The only enhancement here is the prioritization of retransmits as suggested by our model. Figure 4.1 shows the results for the fairness across flows for various configurations over 20 second time-slices. The SFQ implementation is composed of five separate buffers of 5 packets each. The gateway + SFQ consists of five separate buffers each of length 4 packets along with another buffer of size 5 packets for our retransmitted packet queue. The gateway + drop-tail is identical to gateway + SFQ except the 20 packets of buffer space are allocated to a single drop-tail queue. We observe that retransmit prioritizaion improves the fairness of highly shared TCP connections (in the sub-40 Kbps per flow range). SFQ alone performs poorly, likely due to unlucky packet bursts for colliding flows exceeding the 5 packet buffer. We note that this SFQ implementation uses the Click HashSwitch element that does not randomize its hash function regularly that typically helps improve fairness in practice. Interestingly, the retransmit + SFQ allows for the most fairness, but the specific improvement quantities may be an artifact of our bucket and queue sizes. In all cases the total throughput is approximately 99.5% of the link capacity. The main result is that our retransmit prioritization improves upon both SFQ and drop-tail. We observed similar improvements for lower bandwidths across the same range of per flow bandwidth share. We also observed that the fairness significantly improves over longer time scales.

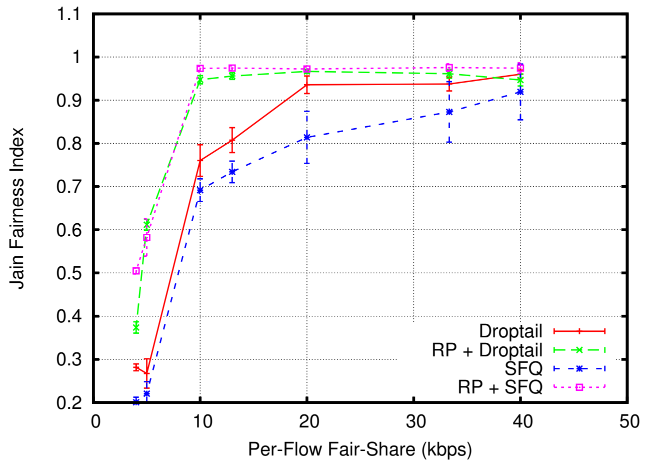


Figure 4.1: Fairness across individual flows over the emulated link for various configurations

Download Rate and Predictability

It is obvious from our implementation that because the user is informed of an admission time, and guaranteed to be allowed to start a flow pool that the system is predictable in terms of the user time of entry into the system. However, a more interesting metric for predictability is the per object download time that directly relates to the quality of service a user should expect after admission. We replay the 2 hour peak load university request log from Figure 3.1. Clients are each configured to open up to four connections at a time, and request objects as soon as possible rather than the logged request time to simulate the dependence of a request on previous requests. Figure 4.2 is a CDF of time taken to download objects of different sizes (10 KB buckets) for the same 1Mbps network configuration as before. We observe that both the download times and overall variance are reduced across the board. The relative improvement for small object sizes is the most dramatic because their short TCP connections suffer the most from being locked out.[2]

[2]We note that the variability in the logs from Figure 3.1 was higher than in our link emulation for a variety of possible reasons including: 1) the actual average RTT to random servers is less than a uniform 200ms as we have in our setup, and 2) the existing proxy likely uses some number of of persistent connections, avoiding connection setup round trips.

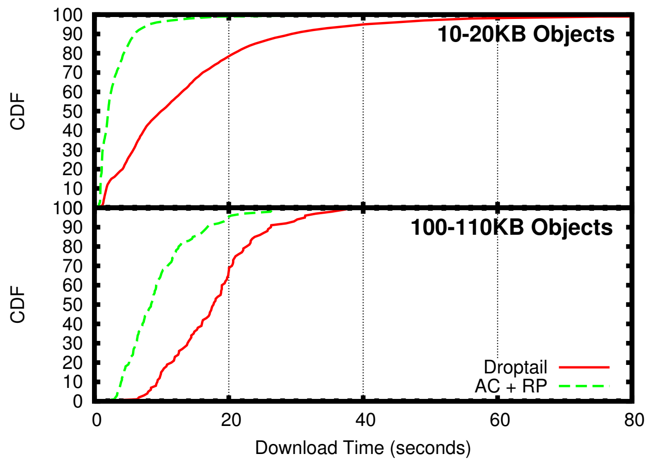


Figure 4.2: Object download times CDF for Admission Control (AC) + Retransmit Priority (RP) and the "baseline" drop-tail queue.

4.5 Related Work

The earliest work on pathological-sharing among TCP flows by Morris appears about a decade ago [192]. We find Morris' solution to be impractical, however; our solution is more practical and deployable. TCP's loss of fairness under pathological sharing is not a surprise, and has been noted in [184, 212, 219]. Work in low bandwidth access links typically assumes a low degree of sharing at the link. For instance, Spring *et al.* [240] and Andrews *et al.* [87] both propose solutions for improving performance at low bandwidth access links, but both operate under low degrees of sharing and assume per-flow fair share of at least 1 packet per RTT. CLAMP [87] uses a middlebox at the bottleneck communicating with receivers to improve TCP performance over low-bandwidth wireless access links. CLAMP implicitly assumes that per-flow fair share is at least 1 packet per RTT, an assumption that does not hold in sub-packet regimes.

4.6 Chapter Summary

In this chapter we detailed several ways to use our TCP model for sub-packet regimes. We arrived at one possible in-network solution to address the issues surrounding TCP collapse. Our middle-box approach uses proxies to perform admission control and fair scheduling across flow pools. This keeps TCP flows within their good operating range without any end-host modification to the protocol. We also extend the good operating region of TCP by prioritizing retransmissions. We showed that our approach dramatically improves TCP across the dimensions of fairness, flow predictability, and object download time.

Chapter 5

ELF: An End-Host Web Optimization Engine

In this chapter, we evaluate the effectiveness of web optimization techniques such as caching stale pages, client-side prefetching, and offline browsing. We show how these web optimizations may be achieved with no modification or permissions beyond installing a simple web browser extension, Event Logger for Firefox (ELF), bundled with our logging tool from Chapter 2. The experiment was conducted at the same school in peri-urban India as Study 1 in Chapter 2 over the same six-week period.[1]

Our high-level web optimization result is that, by aggressively caching and prefetching content, our plugin accelerates the overall web browsing experience by a factor of 2.8x (not counting video content, which appears on less than 1% of browsed pages). We observe an overall cache hit rate of 31%; for pages in which content and all embedded objects are cached, the plugin accelerates load time by 9.1x. While prefetching also proved to be helpful, with 23% of prefetched

[1]Note that prefetched pages were not included in our HTTP traffic analysis in Chapter 2 to avoid contamination of our results.

pages eventually being accessed by the user, prefetching accounted for less than 2% of overall cache hits. We were unable to demonstrate gains due to offline browsing, due to limited network outages and limited depth of prefetched pages. Overall, our findings suggest that a client-side web accelerator can improve the user experience dramatically in slightly constrained network environments such as our peri-urban school.

5.1 Motivation

The technical challenges for establishing Internet access are relatively clear. Beyond improving network infrastructure, many ideas and several systems have been implemented to address connectivity challenges. Solutions exist for problems such as power outages as well as intermittent and last-hop network connectivity [91, 129, 209, 244, 245]. As we show in this thesis, even after a connection is established, connection quality is inadequate; basic problems such as high cost per byte, low bandwidth per person, and high latencies plague even large institutions such as universities that can afford a broadband connection. To make matters worse, modern web pages are designed for dynamic and media-rich content which places greater demands on the network. In conjunction these issues result in an extremely poor user experience in emerging regions, i.e., long and variable page load times, and non-functional websites.

Web acceleration techniques such as caching, compression, and prefetching have been applied in the past to address exactly these bandwidth and latency constraints. More aggressive techniques such as caching stale pages and client-side prefetching have been proposed previously, but only evaluated in disconnected or intermittently connected networks [125, 150]. The main goal of this work is to show that these techniques offer benefits even in connected, but constrained networks. While we do make comparisons regarding the results obtained, we emphasize that the settings are different across several dimensions, including the deployment environment, user population, and task assignment.

5.2 Implementation

In addition to understanding web usage in the context of a peri-urban school in an emerging region, we were interested to explore how much web acceleration techniques recommended by prior work would actually improve the user experience. In addition to logging we implemented several of these techniques in ELF.

5.2.1 Caching without Expiration

The first web acceleration technique that we implemented was to sacrifice freshness for speed and offline availability. The basic idea behind this technique is to allow the presentation of potentially stale cached content to users to reduce the amount of traffic transferred over the network. This idea was recommended 4 years ago [125] and deployed in the C-LINK system in Nicaragua [151]; however, until now it is not been bundled as an easily deployable browser extension. To implement this technique, ELF performs several modifications to Firefox's default behavior:

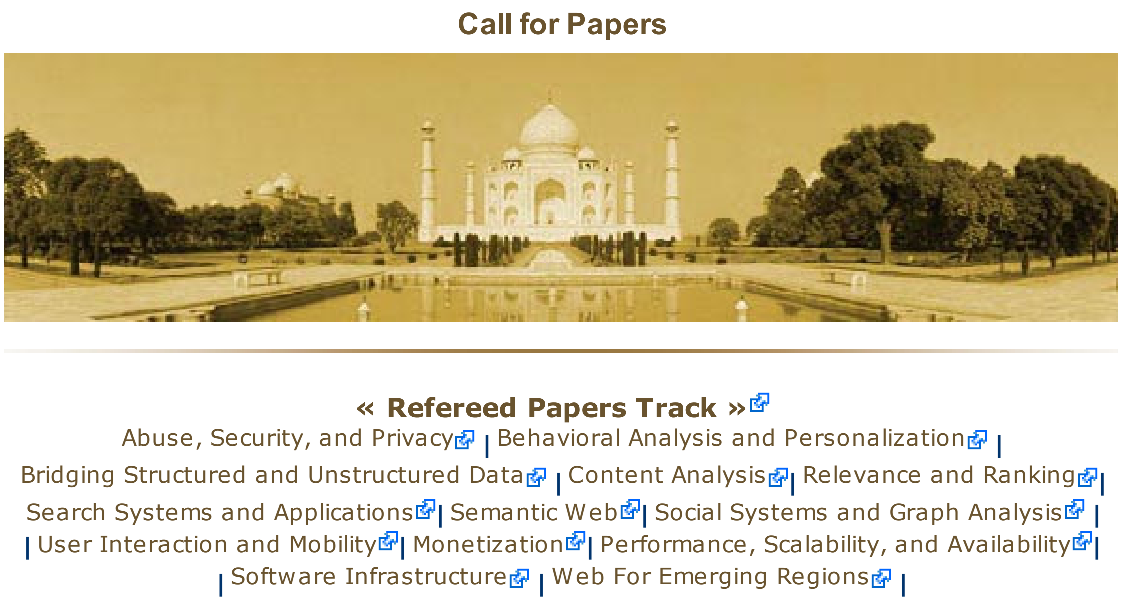
- ELF sets the $check\_doc\_frequency$ to 2 meaning Firefox checks online for an updated version of a file only if it is not in the cache. The Firefox default is 3, which means Firefox checks for an updated file if the cached version is expired.

- ELF changes the cache behavior by extending the cache duration of each file by an extra 30 minutes. The "nocache" headers in all web responses are also ignored and the expiration is set to 30 minutes. While the expiration notices are ignored by ELF (as per the previous point), they are still important for maintaining the correct cache eviction order used by Firefox.

5.2.2 Prefetching

The second acceleration technique that we implemented was prefetching. The state-of-the-art prefetching algorithms described in the research literature attain high accuracies (over 70% of prefetched pages are accessed by the user) [123, 130]. These algorithms typically require deployment at a gateway proxy or server support to achieve these accuracies. Proxies and servers are advantageous because they have more and correlated information to determine browsing patterns. In contrast, off-the-shelf prefetching techniques such as those implemented in WWWOFFLE [76] and Fasterfox [20] perform client-side prefetching, which does not require any additional support. Without access to additional resources, proxy-based and server-initiated prefetching and other proxy-based techniques were not deployable in our setting. Instead, we implemented a simple client-based prefetching algorithm based on an existing prefetching extension named Fasterfox.

The basic prefetching algorithm attempts to download files in anchor links on each page loaded by the user. We extended this algorithm to also keep track of the prefetched frontier of web pages, download their embedded objects, and subsequently linked files as well. These prefetches were performed in breadth first search (BFS) fashion where the files at each depth in the BFS tree were ordered in last in first out (LIFO) order to track the latest potentially useful pages for the user. Finally, prefetching only occurs while the user is not actively downloading any pages. We note that even with our modifications, our client-based prefetching algorithm is substantially less accurate than proxy-based or server-initiated prefetching. While proxy-based and server-initiated prefetching would likely perform better, they require more hardware resources and deployment complexity. The main benefit of our client-side prefetching algorithm is that it is easily to deploy.


Call for Papers

« Refereed Papers Track »

Abuse, Security, and Privacy | Behavioral Analysis and Personalization | Bridging Structured and Unstructured Data | Content Analysis | Relevance and Ranking | Search Systems and Applications | Semantic Web | Social Systems and Graph Analysis | | User Interaction and Mobility | Monetization | Performance, Scalability, and Availability | | Software Infrastructure | Web For Emerging Regions |

Figure 5.1: Offline page rendering with ELF cached link indication (light blue external link icons).

5.2.3 Offline Browsing

The third and final technique that we implemented was offline browsing[2]. While some form of offline browsing has been implemented previously [76, 245] there has not been a formal evaluation of its effectiveness. To support offline browsing, ELF modifies Firefox in the following ways:

- ELF sets the disk cache size to 500 MB uniformly across all machines to increase the amount of offline browsing possible. The default cache size for our version of Firefox was 50 MB. Any existing files in the cache were left in place, and no changes to the other cache size

[2]In a preliminary visit to the school, we discovered that the Internet had been down an entire week due to disconnection somewhere upstream. We were therefore very interested in whether offline browsing could benefit the school.

settings were made.

- ELF modifies the "Cache-Control" header for all web responses in Firefox so that the "max-age=31536000", which indicates that files should be evicted only after they are 1 year old.
- ELF checks the machine is online or offline by sending a HEAD request to a popular search portal every 5 minutes. If the host was unreachable, ELF would put Firefox in offline mode.
- In offline mode, ELF modifies the rendering of all pages using a style sheet to indicate the links that refer to pages that exist in the cache to assist users in avoiding dead links while browsing offline. A screenshot of the offline browsing modification is in Figure 5.1.

5.2.4 Additional Techniques

Other techniques suggested by previous work involving a proxy server were not investigated here due to the lack of access to the gateway proxy at the school. Blacklisting and other filtering was also not implemented (though we easily could have) to avoid contaminating our traffic analysis results. We note that the techniques we implemented are highly synergistic with each other both in terms of purpose and their respective modifications to Firefox.

5.3 Web Optimization Results

While most caching and caching optimizations are simulated or implemented at the gateway proxy, we follow the approach of implementing the caching at the client itself [150, 151]. Consequently, our tool is very easy to install without requiring the user to adjust any proxy settings (as is typical of caching and prefetching systems).

There would undoubtedly be a higher cache hit rate, better prefetching accuracy, and better offline browsing if our techniques were implemented at the gateway proxy. Our numbers should therefore be interpreted as a lower bound for each of these web accelerations if implemented at a proxy. However, our results also represent a realistic measure of what is possible without proxy support that, in our experience, is often the case due to infrastructure constraints, security and privacy concerns, and lack of local technical support. We briefly relate our results to previous findings by Isaacman and Martonosi [151] where it is illuminating, but reiterate that any direct comparison between two different systems in different environments and usage scenarios should be made with care.

5.3.1 Caching

Modifying caching to tradeoff freshness for speed is one of the main web acceleration techniques suggested for constrained networks. We find that with all of the modifications, our overall cache hit rate is 31.1%. The average cache size over the duration of our trace was 110 MB with a standard deviation of 85 MB, a minimum of 0 MB, and a maximum of 383 MB.

Recall that ELF modified the nocache and serves pages from the cache regardless of their expiration time. Compared to Du et al.'s work simulating the aggregate cache hit rate from 6 community information centers the cache hit rate is similar to the unmodified cache rate (approximately 43%) for a cache of the same size (500 MB). Our cache hit rate is relatively high given that the previous simulation was from a trace both 5 years old where dynamic content was less prevalent, and also for aggregate traffic across a set of machines rather than a single client. While we stress that any direct comparison is tenuous, we observe that our cache hit rate 31.1% is higher than that of the local cache-hit rate observed by Isaacman and Martonosi of up to 19%, and lower than their combined collaborative and local cache-hit rate of 78% [151].

Table 5.1 shows the cache hit rate broken down by MIME type. Comparing this table with Table 2.3 we can examine the content types where caching performs well, and those where caching could be improved. We observe that the percentage of cache hits for javascript and

Table 5.1: Cache hits by MIME Type.

Mime Type	Hits	Bytes
images	**35.97%**	24.16%
javascript	34.18%	**34.79%**
html/plaintext	14.77%	4.53%
css stylesheet	8.14%	4.53%
shockwave/flash	4.75%	24.97%
other	2.14%	1.71%
video	0.03%	4.56%
msoffice	0.02%	0.22%
pdf	0.01%	0.36%
audio	$< 0.01\%$	0.17%
compressed	$< 0.01\%$	$< 0.01\%$
binary	$< 0.01\%$	$< 0.01\%$

shockwave/flash are markedly higher than their counterparts for downloaded MIME types. Videos from YouTube and some other sites are not being cached at all resulting in only 4.56% of cache hits being video content. Images and html/plaintext are not cached as often as they are requested, which indicates that the cache could be improved for those content types. In terms of bytes, images and html/plaintext are being cached more than in Table 2.3, but this result is most likely due to inflation caused by the lack of video cache hits.

To measure the impact of caching on the user experience, we compare the load times for URLs that were accessed in both cached and uncached states. As a webpage can contain many embedded objects (such as images), we consider a page to be "completely cached" if both its content and all of its embedded objects are in the cache (this accounts for 35% of total page requests). Likewise, a page is "completely uncached" if neither its content nor any of its embedded objects are in the cache (this accounts for 40% of page requests). In our trace, we identify 716 URLs that were requested in both completely cached and completely uncached states. Comparing the load times for these two states yields an estimate of the performance impact of caching.

Our results are as follows. On average, each page requires 3.3s to load when completely uncached, but 0.36s to load when completely cached. In other words, caching a page and its contents leads to a 9.1x speedup in web access, on average.[3]. Given that the overall cache hit rate for objects is 31%, one could expect caching to offer a proportionate speedup of the user's overall experience: in this school environment, **our tool offers a general speedup of 2.8x across all HTML pages**. We note that this figure does not apply to video content, however, since very few videos are cached; video represents less than 1% of the page requests but 72% of the bytes transferred.

[3]While we presented the speedup using the arithmetic mean, using the geometric mean suggests a comparable speedup of 10.0x.

5.3.2 Prefetching

The number of prefetched files in our trace was $95,714$ files compared to $325,037$ files downloaded by user initiated requests. Despite the aggressive prefetching algorithm we implemented, only 22.7% of all downloaded files were prefetches. Since our prefetcher only downloads files when users are inactive (i.e., when there are no pending requests originating directly from the user), this relationship also indicates that users spent a significant fraction actively waiting for pages to load.

The prefetching accuracy is defined as the percentage of prefetched files that later result in a cache hit. We found the prefetching accuracy in our traces to be 16.13% over the course of six weeks. This figure is low as expected from a single client only prefetching algorithm whereas Isaacman and Martonosi found that with collaborative prefetching achieved 48% accuracy over the course of two days. The percentage of cache hits due to prefetching in our system is only 1.8%. This figure corroborates closely with previously observed results of 2% by Isaacman and Martonosi [151]. These results suggest that while prefetching more pages may improve the percentage of cache hits due to prefetching, the limiting factor to client-side prefetching is still the accuracy of the algorithm.

The prefetched files by MIME type are shown in Table 5.2. Our simple client-based prefetching algorithm only downloads the links on the pages viewed by the user, and only after all links on the page are retrieved are the embedded objects on the linked pages downloaded. As new page requests from the user come in, new links are added, and the user likely browses too fast for the prefetching algorithm to download all of the links and start downloading the embedded objects on the linked pages.

Our results suggest that if more bandwidth were available or a more intelligent algorithm were employed, it would be beneficial to prefetch images and javascript files. More advanced

Table 5.2: Prefetched Files by MIME Type.

Mime Type	Hits	Bytes
html/plaintext	**93.22%**	**55.36%**
images	2.85%	7.58%
other	2.66%	1.90%
pdf	0.62%	17.06%
shockwave/flash	0.58%	0.33%
audio	0.04%	9.22%
msoffice	0.03%	0.69%
video	$< 0.01\%$	2.67%
compressed	$< 0.01\%$	5.19%
javascript	0.00%	0.00%
css stylesheet	0.00%	0.00%
binary	0.00%	0.00%

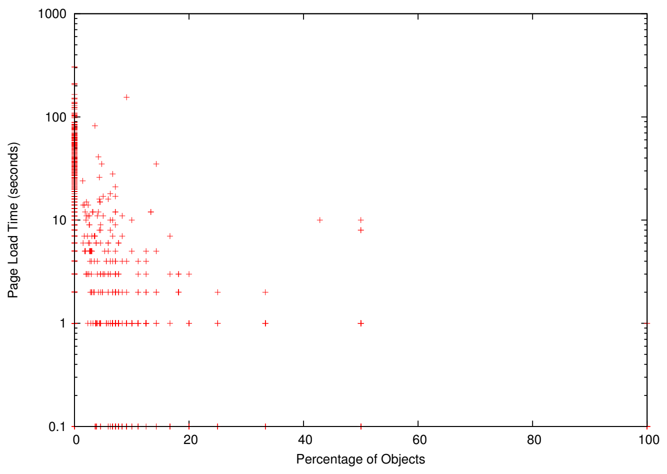


Figure 5.2: Page load time versus per-page prefetch rate (fraction of requests that hit in the cache and the file was cached due to a prefetching event).

prefetching algorithms are outside the scope of this work, but one simple improvement for future work would be to estimate the likelihood that a user will follow a particular link on a page, and to prefetch content from the link with a priority that is in proportion to that likelihood.

As with our caching results, what we are really interested in is the impact of prefetching on the user experience. Unfortunately we do not have enough data points to do a rigorous analysis of completely uncached versus completely cached (and prefetched) pages, though we illustrate the general relationship between load time and percentage of prefetched embedded objects in in Figure 5.2. We can observe from these results that prefetching is helpful if files are chosen properly.

5.3.3 Offline Browsing

While our original impression was that the network connectivity was frequently down in the school, our logs indicated very few periods when the browser was used in offline mode. We did log a negligible number of cache hits (12) during offline browsing, though only 8 machines were used while offline, for a total of two hours of logged outages. We intentionally did not train students and teachers regarding offline mode, so they might not have attempted to browse cached pages while the network was down. However, even if they had attempted to use the browser, with the current rate of prefetching they could expect to follow at most 1 or 2 links before requesting pages that were unavailable.[4] We hope to revisit offline browsing in other environments in future work.

5.4 Related Work

The work most closely related to ours is that of Isaacman and Martonosi, who propose aggressive techniques such as caching stale pages and client-side prefetching and evaluate them on network traces [151] as well as via a real deployment in Nicaragua [150]. While their focus is on "collaborative caching" that allow multiple clients to share cached pages, their techniques are similar to ours when restricted to a single machine. One of the primary differences between our works is the deployment environment: while we target a synchronous Internet connection, they deploy in an asynchronous delay tolerant network (DTN) [129]. We also improve the ease of installation and deployment by packaging our system as a browser plugin, rather than requiring changes to the proxy settings. Finally, our deployment incorporates data from a longer period (six weeks as opposed to five days [151]), enabling detailed analysis.

A wide variety of caching algorithms exist [220,256], and recent research targeting emerging regions has looked at caching architectures for affordable hardware [90]. Tangentially related to our work is Opera Mini, a browser designed to accelerate browsing for mobile devices and focuses on page rewriting and compression and require an additional proxy component [51]. Prefetching systems are also well studied, but are also dated and untested in this context and

[4]Isaacman and Martonosi make exactly this observation in their often offline C-LINK deployment.

require access to a proxy or server support [126,130,202]. Furthermore, emerging region specific ideas such as proxying, text-only browsing, offline browsing, and collaborative caching have been suggested and implemented but specialized systems are often difficult to deploy and scale up because they require additional resources or expertise [37, 76, 149, 152, 226, 245].

Lower in the networking stack, transport or session layer protocols such as SCTP [56], SST [59], and SPDY [58] have also been implemented and experimented with, but they are designed as performance optimizations for high speed Internet connections. Internet statistics reports from sources such as Akamai [4] and the International Telecommunications Union (ITU) [30] indicate that, for a variety of reasons, broadband growth in many areas of the world including most of Africa and South America continues to lag behind. As a result, optimizations that focus on slow Internet connections will continue to be relevant, particularly if the optimizations are deployable with low levels of expertise. Finally, web acceleration techniques that are browser based have been implemented and deployed extensively [20, 27, 123], but to our knowledge no formal study in any emerging-region context has been conducted to evaluate how well these techniques actually perform.

5.5 Discussion

In this section we briefly discuss some of the implications of our results.

From our results, freshness vs speed is a good tradeoff for users behind constrained network connections. We realize that this tradeoff violates RFC2616 [132], but, as suggested by prior work, users in these settings are unlikely to be interested in minute changes in content on sites other than news, webmail, and Facebook [125]. We also found that prefetching, while beneficial, could be further improved, and client-end prefetching algorithms are an interesting avenue for future work. We found that offline cache browsing is only useful for visiting previously viewed pages. While some prior work has suggested that less experienced web users perform history-based browsing [221], this appears to be a corner case in our setting. We hypothesize that another shortcoming of offline browsing is that the rate pages are prefetched or cached cannot keep up with the pages browsed while offline. Under a constrained setting, a user will eventually get a cache miss and fail to make progress. Local search and offline content aggregation are promising options to address this shortcoming of offline browsing as we see in Chapters 6 and 7. Also, we still consider highlighting dead links as a useful user interface feature given the caveats above.

There are several other optimizations that we considered, but were unable to or did not wish to implement in this work to avoid contaminating our traffic results. Time shifting (e.g., downloading large volumes of content overnight for possible viewing the next day) is not feasible in this setting due to limits on monthly bandwidth consumption. If bandwidth were not a concern, time shifting the prefetches or downloading even while user requests were taking place could be helpful. Text-based browsing requires a proxy server to pre-render pages prior to transferring them over the bottleneck link. The client could potentially interface with such a proxy, and this is an area for future work. Compression requires server support, which is unlikely to be ubiquitous until web server software comes pre-configured to use compression. Caching of dynamic content would be tremendously useful for offline browsing and improving cache hit rate, particularly with our caching modifications. In the research literature, existing dynamic web caches exist, but they are primarily focused on server-side or proxy-based implementations. Leveraging the existing implementation in most browsers to "save webpage" could be one easy solution. Blacklisting of advertisement files would improve performance, but filtering pages would have severely contaminated our traffic results. We will be incorporating or deploying the adblock extension in the near future [2].

Finally, there were two optimizations that could have been useful that we did not implement, but have been studied independently. Collaborative caching across networks behind upstream bottlenecks such as ours could have been a way to gain the greater benefits of caching without access to the gateway proxy [150]. Caching partially downloaded or chunks of content could also be useful particularly for YouTube videos and highly synergistic with cooperative caching. We hope to add these caching optimizations to our extension in the next version. While the prevalence of video may appear to undermine the benefits of our optimizations, we note that the video downloads are too slow to be viewed in real-time. This suggests that people are loading the videos first and watching them after completion, and in terms of user experience, these load times are not as disruptive as those during a more real-time task such as web browsing.

On the clients themselves, browser modifications are an ideal choice for the many deployment constraints in emerging regions. Beyond the deployment and scaling issues we have discussed in Section 5.1, awareness of the solution itself is also a problem. Even after the extension is made available for free download it is unlikely to be discovered by a local sysadmin who does not read research papers! To overcome this obstacle, the extension we implemented in this work would ideally be integrated into the browser itself so that the software comes pre-packaged with constrained network web optimizations. To address the lack of user expertise, the browser could track page load times and turn on the various web optimizations as the user experience starts to degrade. Users of the browser with good connections or tech savvy users who elect to turn off auto-tuning would not be affected.

We do not claim that our extension completely solves all of the web access issues behind constrained networks. In fact, we believe the opposite: the optimizations here cover a wide variety of options deployable at clients and is an important first step for slightly congested network conditions. Our results and work in the following chapters indicate that the optimizations requiring gateway proxy support are likely to be crucial. While the changes we make to caching and prefetching violate several existing web standards, we believe that the benefits far outweigh the shortcomings in these situations. We hope that our work encourages a more systematic adoption of these techniques.

5.6 Chapter Summary

In this chapter, we implemented many of the web accelerations suggested previously as a Firefox plugin – freely available and easily installed on any client machine. We found that serving stale pages works well in practice and prefetching helps to some extent, though offline browsing was not useful in the context of the school. We quantified the benefits offered by our tool via a six-week deployment, demonstrating a cache hit rate of 31% and accelerated viewing of cached pages by 9.1x. Taken together, this implies an average acceleration of 2.8x for the user in browsing (non-video) web pages. We found that many more complex optimizations suggested by prior work are challenging due to the lack of server support, local expertise, and permissions from the school.

Chapter 6

RuralCafe: An Asynchronous Model for Web Search and Browsing

As we have seen in the previous chapters, there are limitations to how good optimizations can maintain good user experience both at the application layer and network layer. At some point, interactive applications such as web search and browsing are too frustrating or simply do not work. In this chapter, we present the design and implementation of *RuralCafe*, a system intended to support efficient web interactions over intermittent networks. We use the term web *interactions* or *browsing* as a shorthand in this chapter for the combination of web search and browsing activities that people engage in while using the web.

RuralCafe enables users to perform web interactions asynchronously and find what they are looking for in *one round of communication* as opposed to multiple rounds of iteration. RuralCafe does this by providing an expanded query interface that allows a user to specify additional terms and constraints to maximize the utility of the results returned per round. Given knowledge of the limited available network resources, RuralCafe performs optimizations to prefetch pages to best satisfy a query based on a user's preferences. We describe the design of RuralCafe and evaluate it using queries from logs made to a large search engine, queries made by users in an intermittent setting, and live queries from a small test deployment. Finally, we compare two different web interaction models: the conventional web model and an asynchronous model in a detailed user study at one pilot in rural India.

6.1 Motivation

Existing work on bringing the web to emerging regions has generally focused on addressing either system level infrastructure issues or specialized user interfaces (e.g. One-handed thumb use on small devices) [163, 172, 174]. However, research that focuses on constrained web access is either dated and less relevant due to changes in the web [180, 202] or, in the case of more recent work, is in the context of mobile devices [101, 229, 252]. While there is extensive literature about web use in general, until very recently little effort has been directed at understanding the unique characteristics of Internet use in emerging regions [125, 148, 251].

A few network constrained applications have been designed and implemented by splitting the application into a fully synchronous front end and an asynchronous back end [182, 245, 251]. Specifically, the Time Equals Knowledge project (TEK) allows queueing of pages for asynchronous download over simple mail transfer protocol (SMTP), and local search functionality. However, the effects of applying this asynchronous model of interaction and common web optimization techniques on users have not been formally evaluated. RuralCafe's design was inspired by TEK.

To motivate the design of RuralCafe, we will first describe different web browsing scenarios in emerging regions and how conventional browsing mechanisms are unfit for these environments. We then describe a recent user study we conducted in a large university in India where nearly 400 students simultaneously share a low-bandwidth Internet link. The primary result from the study is that the traditional style of web search is unfit for very low-bandwidth and intermittent environments.

6.1.1 Constrained Web Browsing Scenarios

Shared Budget-constrained Low Bandwidth Networks: Many users ($50 - 1000$) in an institution share a connection to the Internet using a low-bandwidth (e.g. 128 Kbps) link with a pay per use model. This scenario is common at university and small businesses in emerging regions around the world. One example is the university network we analyzed in Chapter 2. Another common example are the rural Business Process Outsourcing (BPO) units [55] in India that consist of $50-100$ people sharing a 64 Kbps Internet connection. The cost of these links is relatively high and is either determined by the total usage time or number of bytes transferred. As we have seen in previous chapters, even when connectivity is available, the available bandwidth per user is often low. From the perspective of the user, these networks appear intermittent (though the period of intermittency is short, e.g. on the order of a few minutes). To partially alleviate the low-bandwidth problem, one approach is to use admission control policies to provide a more interactive experience to a few users, but then the network appears intermittent from the user's perspective.

Kiosks using Mechanical Backhauls: United Villages [67, 214], a for-profit organization has deployed WiFi-enabled kiosks in various villages in Asia, Africa and Latin America. Each kiosk uses a mechanical backhaul link based on physical transportation systems such as buses are used to transport bits between the village and the closest urban area. These links have long delays and are operational only a few times every day (often once), but can transfer gigabytes of data in one trip. To make kiosks usable, each kiosk has an operator who acts as an interpreter to help users surf the web.

Search using Messaging Links: Given the penetration of cellphones in rural regions, it is possible to envision a scenario where a third-party (or the cellphone provider) provides cellphone users with the ability to perform web search queries using SMS or the multimedia messaging service (MMS). One reason for providing a messaging based search service is that in many rural regions, messaging is significantly cheaper than voice calls or data service. With such a service, the user could limit costs of the service based on their search needs. An SMS message contains up to 140 bytes whereas MMS messages can return several KBs of data in a single message.

Several points are noteworthy from these scenarios. First, the user is aware of the intermittent network and is also willing to wait correspondingly for a response. Second, the nature of the intermittent links in each of these cases varies. Mechanical backhauls are high latency links with the potential for bulk data transfer, but the other two cases exhibit low latencies and low bandwidths. In the budget constrained case, network connectivity is available, but its use is limited.

6.2 RuralCafe Design

The conventional search process using standard web browsers proceeds as follows. First, a person issues a query to the search engine and then the server returns a search results page. This is one ’round’ trip over the network. Then a person looks at the results and if he does not find the desired information he reformulates his query and requests new results. This is another round. If the person finds a page they are interested in, he clicks on the link requesting that page from the web server. The web server then returns the page and that is another round. If the page has embedded objects such as images or scripts, the browser automatically requests the embedded objects from the servers that are hosting those pages resulting in a fourth round of communication.

The central theme of the RuralCafe design is to rethink the search process in the context of intermittent networks. The conventional model of interactive web search and browsing just described is not suitable for intermittent settings. As connections get slower, more round trips means the user experience degrades further, and the user has no recourse to ensure that server responses suit his needs. To overcome this problem, RuralCafe tries to maximize the utility of search response for every round of search. To achieve this goal, RuralCafe focuses on four design principles:

1. Rethink the Search Interface: The traditional search interface is not expressive enough for intermittent search. RuralCafe uses a modified search interface that explicitly exposes the underlying network intermittency to the users and enables users to express their search intent and requirements in a greater level of detail. Part of the challenge is also to make the search interface relatively simple, since users typically do not prefer complicated search interfaces. In addition, any redesign of the search interface should not require any modifications to the browser software at the end hosts.

2. Local Query Refinement and Search: Many user queries are often ill-specified and ambiguous. In contrast, to maximize the utility of a search response, we expect every query input by a user should be meaningful and as well specified as possible. One way we address this issue is to ask users to explicitly specify as part of the search interface if their search query is well-specified or not (deep or broad). Another way we deal with this issue is to perform local query refinement and localized search. RuralCafe uses a large local cache that is pre-populated and enables users to exhaustively search the local cache before issuing an intermittent search query. Local search also enables users to appropriately refine their query to reduce the number of search rounds. RuralCafe uses a repository of "popular search phrases" to support query expansion and also correct potential errors in queries.

3. Adapting to different Intermittent Environments: RuralCafe uses an intermittent link model to encapsulate different types of intermittent networks under a common set of parameters. This enables RuralCafe to easily adapt across different environments and also quantify the available network resources for each query.

4. Search Response Personalization: RuralCafe tailors the search response for a query as a function of the user search preferences and available network resources. Depending on the type of query, RuralCafe prefetches an appropriate set of related pages to a query that enables users to locally search. In addition, RuralCafe employs different filtering and compression routines to prefetch several pages within the available quota for a query.

6.3 RuralCafe Architecture

In this section, we describe the RuralCafe system architecture. We begin by describing the RuralCafe setup and query process. We then elaborate on three key components of RuralCafe: (a) local search and query refinement; (b) search response personalization; (c) adapting to different intermittent environments.

6.3.1 RuralCafe Setup and Query Process

RuralCafe uses a simple proxy architecture(Figure 6.1) comprising of end-hosts within a village that connect to the Internet over an intermittent link using intelligent proxies. One proxy is placed at either end of the intermittent link. All the traffic to and from the end-hosts traverse the local proxy before being injected on the intermittent link and tunneled to the remote proxy. The remote proxy connects to the Internet using a direct network connection.

The local proxy is equipped with a large local store that the client can locally search without using the network. When user directed requests require the network they are queued and dis-

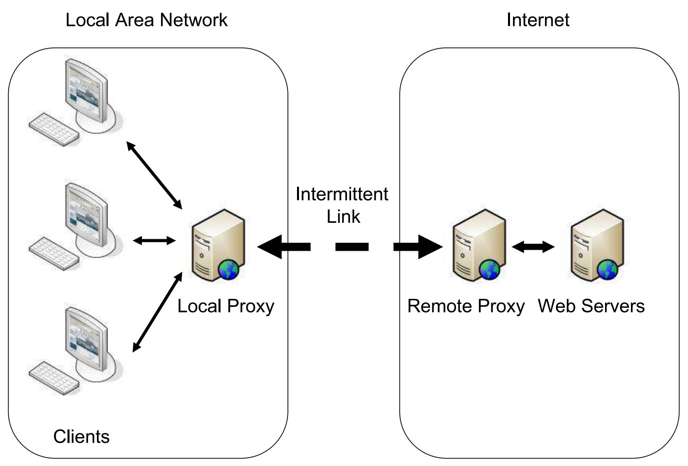


Figure 6.1: Basic RuralCafe proxy architecture.

patched to the remote proxy. The remote proxy continually prefetches pages for return to the local proxy opportunistically.

In practice, the intermittent link could be a multi-hop delay tolerant network with the intelligent proxies placed at the edge of the DTN. If a single end-host directly connects through an intermittent link, as in the case of cellphones, then the local proxy functionality is placed at the end-host. End-hosts are configured to connect to the local proxy by default. The local proxy redirects the user to the expanded search interface when a search engine is requested. When possible, the local proxy is equipped with a large local store that the client can locally search. The entire RuralCafe functionality is deployed at the two proxies.

The user's browser home page is first set to *www.ruralcafe.net*. The user requests the RuralCafe portal page from the local proxy. The portal page is served directly by the local proxy and contains the RuralCafe user interface that we now describe in detail.

6.3.2 Local Query Refinement and Search

We populate the local proxy with a subset of the popular N-grams from the Linguistic Data Consortium (LDC) [36]. This dataset provides a large corpus of N-grams along with their web frequencies as published by popular search engines. We use the popular LDC dataset to perform two simple optimizations to assist in offline query formulation. The first is to suggest associated popular query terms to the user corresponding to a search query. RuralCafe, then allows the users to choose appropriate query expansion terms from a list of popular terms. The second optimization is to correct potential grammatical errors on individual terms and provide alternatives to these terms to the user.

Each page cached by the local proxy is associated with a set of terms that includes the list of query terms fetched, the page, and a list of important terms in the document. Our current implementation fetches the important keywords in a document including the titles, search keywords, names, section headings and keywords with references. We explicitly wanted to restrict the number of terms associated with each page especially since we did not want to impose processing overhead on the local proxy. The list of terms in a new query is compared with the list of terms associated with each document and the local search similarity is simply a match of the number of common words above a certain threshold (we use a threshold of 2 for multi-term searches and a threshold of 1 for single-term searches). In the future, we intend to support more elaborate local search features (by mining the terms of the downloaded documents) to enhance the search capabilities during disconnected periods.

6.3.3 Query Response Personalization

The local proxy forwards the expanded search along with the query options to the remote proxy. The local proxy in RuralCafe associates each query with a *search quota* that represents the maximum number of bytes that is allocated as the overall response budget for a query. This quota is calculated based on the available network resources and the number of outstanding requests. Alternatively, this quota may be set by the system administrator. Generating the response to a query at the remote proxy involves two steps: (a) issuing the appropriate search query to a search engine to gather search responses; (b) determining what information to be pre-fetched within the allocated quota for a query. We will now elaborate on these two steps.

Processing a Query: RuralCafe currently supports two different forms of user queries: *simple queries* and *contextual* queries. A simple query Q is just a collection of terms and this query is forwarded to the search engine as constructed. This is the default query format of RuralCafe. A contextual query is specified using a keyword W (within a known set of keywords) representing a specific topic that drives a specific automated filtering or extraction process at the remote proxy. Contextual queries are useful for providing information for the remote proxy to narrow down results. We discuss contextual search in detail in Chapter 10, and show how contextual search queries are useful for SMS-based search where the search engine response is limited to only 140 bytes.

Customizing the response: For a given query Q, the results gathered by the remote proxy can be classified into three basic categories: (a) Search results; (b) Downloaded or prefetched pages; (c) Embedded media content. The search results refer to the top-M search results along with brief summaries. Downloaded pages refer to the basic files associated with web pages (HTML of various flavors, CSS, ASP, etc.) present in the search result pages. Embedded media content refer to the embedded files (images, audio, video, and other unidentifiable files) in the page.

The overall search quota for the query Q is divided between these categories depending on the *richness* and *broad vs deep* parameters. For a broad query, a larger portion of the quota is allocated to search results and a smaller portion for downloaded pages and embedded content. For a deep query, more of the quota is allocated to downloaded pages. The richness preference of a query response is dictated by the user preference, but ultimately limited by the available quota for a query. For “text only” queries, we extract only the individual links on search results pages, excluding the various cached links, link previews, advertisements, and other page content. Similarly, for downloaded pages, we do not include the return of embedded media content for each page. Only the bare-bone page HTML for the downloaded pages are prefetched. For “everything” queries, we allow embedded images, audio, video, and other unidentifiable files to be returned without filtering.

6.3.4 Intermittent Network Adaptation

We require RuralCafe to work seamlessly across a variety of different intermittent networks. To achieve this, we use a simple model that can be used to parameterize different types of intermittent links. based on the *bundle* concept from the Delay-Tolerant Networking (DTN) architecture [129]. Transmissions across a link are batched into bundles where within each bundle we can pack information up to a pre-specified maximum bundle size s. Each bundle upon transmission is associated with a delay d and bundle transmissions are separated by time-periods T. To model budget constrained links, we use a parameter N to represent the maximum number of bundles that can be transmitted within one day. N and T are related in that one parameter imposes a constraint on the other. Typically for an intermittent link, we would use one of the two parameters as the primary parameter to determine the number and time between bundle transmissions. In essence, we associate every intermittent link with four parameters: (a) bundle size s; (b) delay d; (c) time-period T; (d) maximum number of bundles N. While we assume s to be a fixed value, d and T can be variable. We use the value of s to determine the search quota and the values of d, T, N to calculate approximate response times.

Estimating Search Quota and Response Time

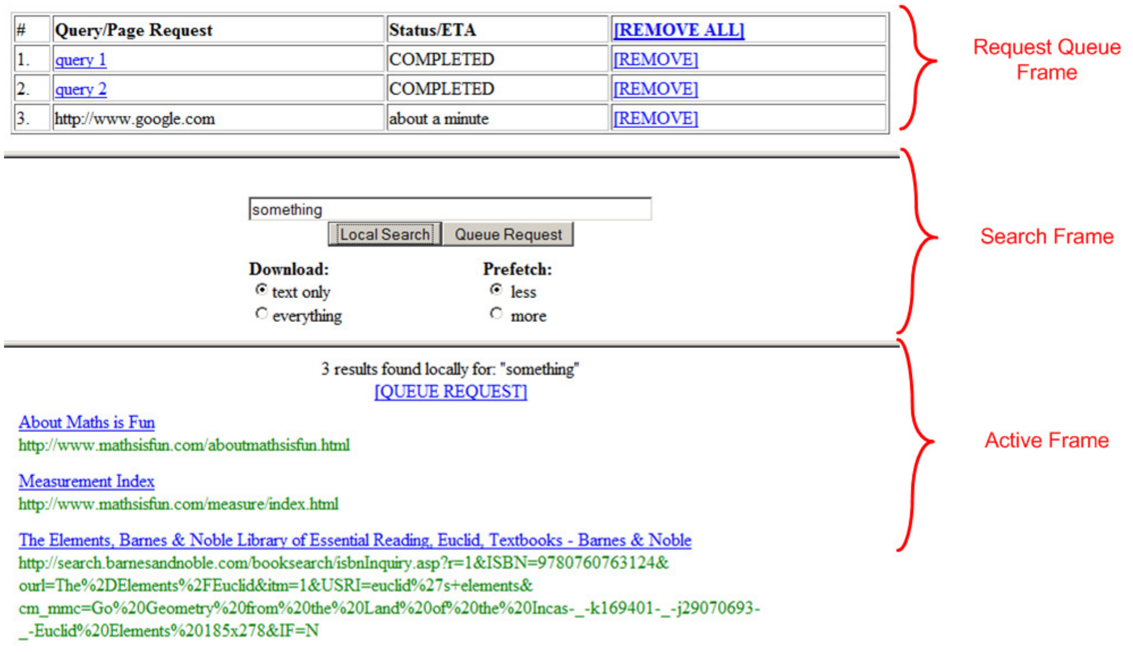
The local proxy batches queries into "sessions" where a session is a collection of queries issued together within a bundle. Hence, the net quota for a session is s. In mechanical backhaul networks, a session is purely determined by the arrival times of the physical transportation mechanism; here, a session is a set of queries issued between two arrival events. In budget-constrained links, a session is dependent on N, the maximum number of bundles allowed per day and the arrival rate of user queries. Ideally, one would wait until one gathers a constant number of queries (determined using N and mean arrival rate) and issue them together in a session. RuralCafe does not make distinctions across different queries. Given a session with K queries, RuralCafe assigns a common quota of s/K for every query for a bundle size s for the link. Note that in the case of SMS or MMS based messaging links, we treat each query separately and set s to be the maximum allowable message size in the underlying network. We calculate the response time as a linear function of the existing queue size and the parameters d, T and N.

6.3.5 RuralCafe Interface: An Asynchronous Queuing Model

In this section, we describe the asynchronous queueing model of the RuralCafe user interface.[1]

The interface of our web search system consists of three components presented to the user in the form of browser frames: *Search Frame*, *Request Queue Frame*, and *Active Frame*. The user interacts with each of these frames as illustrated in Figure 6.2.

[1]RuralCafe's user interface has gone through several iterations since its initial design in 2008. The user interface features here are from the most recently released version of our system (2010). Many of the features present in previous versions of RuralCafe have been subsumed into system configuration settings.



#	Query/Page Request	Status/ETA	[REMOVE ALL]
1.	query 1	COMPLETED	[REMOVE]
2.	query 2	COMPLETED	[REMOVE]
3.	http://www.google.com	about a minute	[REMOVE]

something

Download: [✓ text only] everything

Prefetch: [✓ less] more

3 results found locally for: "something"
[QUEUE REQUEST]

About Maths is Fun
http://www.mathsisfun.com/aboutmathsisfun.html

Measurement Index
http://www.mathsisfun.com/measure/index.html

The Elements, Barnes & Noble Library of Essential Reading, Euclid, Textbooks - Barnes & Noble
http://search.barnesandnoble.com/booksearch/isbnInquiry.asp?r=1&ISBN=9780760763124&ourl=The%2DElements%2FEuclid&itm=1&USRI=euclid%27s+elements&cm_mmc=Go%20Geometry%20from%20the%20Land%20of%20the%20Incas-_-k169401-_-j29070693-_-Euclid%20Elements%20185x278&IF=N

Figure 6.2: RuralCafe user interface. The three frames are labeled in red. The user is free to interact with any of the frames at any time.

Local Search and Query Refinement

Local search assists the user in offline query refinement by allowing the user to search through the cache at the local proxy. In addition, offline query construction assistance is provided in the form of suggested queries similar to those of major search engines. Currently, suggested queries are returned by the local proxy (using a simple term frequency database), which also does not require any network resources.

The user is presented with a simple *Search* frame to perform local searches and query refinement. We made the design decision to dedicate screen real estate to the search frame to encourage the use of local search, which we believed would improve the usability of the feature. Requests made in this frame may either be served locally without use of the network or added to the queue in the *Request Queue* frame. Once a local search is performed, the local proxy returns a list of links to pages in the cache along with suggested queries (Suggested Queries & Local Results in Figure 6.2). The user may click on the links to view the pages or click the suggestions for another local search. The user could also queue the query for download if the local results were unsatisfactory.

Queueing

The main difference in the asynchronous model is the use of queueing to decouple user requests with network availability. What this means is that for pages that are not in the local proxy's cache (require network access), the user can queue up the page for download and return to it after it is downloaded. While the server processes the queue, the user can continue to perform other searches or queue more page requests. All requests that require network access must be added to this queue.

The *Request Queue* frame in Figure 6.2 displays the list of pages queued by the user and list of the queued items and their status for browsing. If a page is being downloaded or waiting in the queue, its associated expected completion time is displayed. If a page in the queue is completed, a link is displayed to access the page. If the link is clicked, the results are served locally and presented in the *Active* frame.

The user is informed of the pages that must be added to the queue and then is free to do so if the page is desired. The user is free to add or remove pages from their queue at any time. We dedicated space in our user interface to display the queue so users could always see the status of their requests and act upon the information easily.

Remote Requests and Prefetching

The *Search* frame also allows users to queue up remote requests. The interface is a search box and two buttons for queueing either a search or a URL to be downloaded.

The interface requires two additional parameters from the user: *richness* and *depth* to direct the search process. The *richness* parameter is presented to the user as "download" radio buttons to select either "text only" or "everything" to download. The *depth* parameter is presented as "prefetch" radio buttons to select "less" or "more" pages. The default is for text only and prefetching depth of one link.

Internally, RuralCafe defines other parameters for each query based on configuration settings and algorithms. These parameters are not exposed to the user interface by default. Each request is associated with a *quota* that represents the maximum number of bytes that is allocated as the overall response budget for a query. Queries are also categorized as "deep" (well-specified) or "broad" (not well-specified). The effect of deep versus broad queries is to determine how RuralCafe selectively prefetches search response pages. With the parameters associated with each query, the remote proxy prefetches an overall response to a page request recursively until a

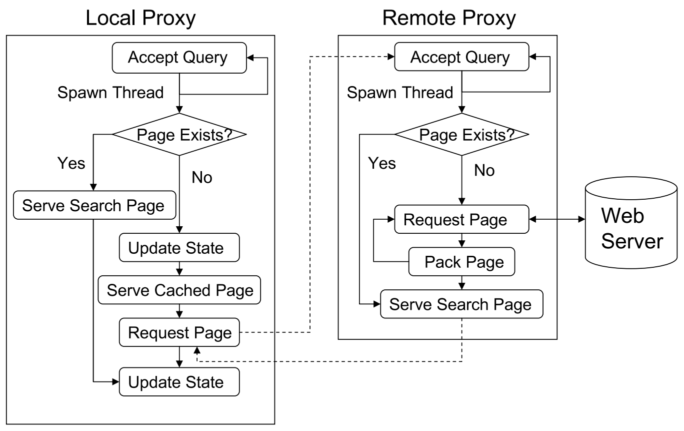


Figure 6.3: Implementation overview.

quota is met or the depth limit is reached. Once complete, all downloaded pages are returned to the local proxy to be incorporated into its cache.

6.4 Implementation Details

In this section, we will describe the implementation details of RuralCafe. The RuralCafe prototype is implemented in 10000 lines of C# code. Figure 6.3 illustrates the overview of the proxy implementation of RuralCafe. The local proxy and remote proxies similar share many of the same components. Both proxies are multi-threaded and cache results on local storage. The client browser is located on the same LAN as the local proxy and is configured to use it as the proxy server.

6.4.1 Local Proxy

The two responsibilities of the local proxy are to service page requests, and to give estimates on how long the pages will be returned.

The local proxy listens for connections on port 8080 from users and spawns a new thread to service each HTTP request. Each request is served only from the local cache. The local proxy maintains a queue of requests per client IP address; the thread adds the latest request into the this list, and time-stamping it with the request time. The thread then serves the requested page from the cache if it exists, otherwise it returns a the default search page.

After the thread serves the default page, it forwards the request out to the remote proxy and awaits delivery of the requested page. The client's browser is forwarded back to the default search page. Once the requested page is returned, the local proxy unpacks the page into the local cache and updates the status of the request in its internal state, and the thread is destroyed. The next time the client requests the page either directly or via the link presented on the default search page, it will be served immediately from local cache. The local proxy is also stateful, so that if a user logs out of the machine and then returns later his queries will have made progress, and returned results are available for browsing.

The local proxy communicates the quota allocated for the query to the remote server, along with the preferences associated with the search or URL request. The request parameters set by the user are simply attached to the search URL by the web form to be parsed by the remote proxy.

6.4.2 Remote Proxy

The primary role of the remote proxy is to service requests made by the local proxy, to prefetch pages filtering out page components according to search request preferences, and finally to return the results to the local proxy. Unlike the local proxy, the remote proxy maintains much less state and is a prefetching agent that speaks the same protocol across the intermittent link as the local proxy and fetches query results on its behalf.

Once the remote proxy receives a request it spawns a new thread to serve the request. First, the remote proxy parses in the URL for the parameters and preferences of the request (quota, type, richness). Then the proxy checks its cache for the page, and requests a fresh copy if necessary. If a page needs to be downloaded, the proxy awaits a response just like a normal web proxy. After the proxy receives the page from the server, it stores the page in its cache. Pages are stored locally to simplify the composition of content to be sent back to the local proxy. The remote proxy then creates a single archive to store the results of the request. If the page requested was a search page, the proxy follows the policy settings given by the local proxy and prefetches the appropriate number of search results. The proxy then fills up the remaining space with the pages in the search results as best it can according to the breadth and richness setting of the request. If the quota indicates that there is extra space after this, the proxy continues to fill in pages and/or embedded elements until the quota is filled. Finally, once the quota is full, the remote proxy sends the result back via a HTTP Response headers to the local proxy including a special header "encoding-type=gzip-package" to indicate that it is a gzip RuralCafe response.

Prefetching and Prioritization Policies

There are clearly many alternative policies for prefetching pages including: filtering and condensing content, or recursively retrieving page links. There are also more sophisticated prefetching algorithms available [220], but these are beyond the scope of this work.

The remote proxy is also in a position to make decisions about the prioritization between different queries for the link bandwidth back to the local proxy. Our current implementation supports only a simple FIFO allocation where the responses to requests are returned according to when they have been successfully fetched. In the future, we plan to experiment with other allocation strategies based on number of query terms, size of the downloaded search results, differentiating across different types of link characteristics. It is however, important to keep the algorithms the user has control over relatively intuitive so that they have a sense of what to expect from each of the different search settings and parameters.

6.5 Microbenchmarks

It is difficult to completely evaluate the benefits of RuralCafe without a detailed user study. We have recently deployed RuralCafe in a large university in rural India, and plan on conducting such a study in the near future. Currently, we have conducted a needs-assessment user study, and performed simulations, benchmarks, and a usability study in a testbed in our lab. In this section, we present the results from this evaluation of RuralCafe. We evaluate the efficacy of RuralCafe across two metrics: (a)time saved and (b) adaptability to different network constraints. The analysis for usability and reduction in search rounds is summarized, whereas the results on RuralCafe's adaptability to different network constraints is shown in more detail. In our simulations and benchmarks we use search query logs from search queries collected in an Internet cafe in Cambodia used in a study by Du at al. [125].

6.5.1 Idle Time Reduced

The amount of time saved is clearly dependent the connection quality to the Internet; the slower the connection, the more time is saved by minimizing search rounds over the Internet. While the connection quality varies with the setting, the number of search rounds does not. To get a lower bound on the number of search rounds are in a typical search session we performed simulations using our search logs and compared this with previous results. In these simulations we aggregate the search queries into search sessions by checking for matching terms across neighboring queries. A search session is indicative of the number of rounds of interaction of a user for a single search. Our results showed that roughly 80% of the queries were present in search sessions that were not satisfied in a single round (i.e. more than one query per session). In fact more than 30% of the search sessions involve at least four queries. In a larger study from AOL [207], Pass et al. found that 28% of queries are re-formulations of the previous query, and that in these cases the query is reformulated an average of 2.6 times. In the intermittent world, searches that require more than a single round may increase the time for meaningful results by a factor of minutes, hours, and even days. This is a promising indication that providing an interface that gives the user results in one round would increase the practicality of one-round web search.

In our testbed the most common feedback was that users spent more time navigating the suggested search terms along with the cached search results to come up with a more clearly specified query. However, after adding the desired query to the queue, users felt comfortable leaving the search to run and coming back at a later time. While it is true that query construction assistance and local search of the cache sometimes causes users to perform more total search rounds per session, by the time the search query actually uses the network the query is so well refined that often only a single round is necessary.

6.5.2 Adaptability to Different Networks

Prefetching Flexibility: Using the search logs available to us, we wanted to explore whether RuralCafe could offer a gradation of services for variable quota levels ranging from over 1 MB (backhauls) to only 100 KB (dialup). We found that for larger quota sizes, RuralCafe can prefetch several search result pages as part of the search response to significantly enhance the subsequent local browsing experience. For small quota sizes, the preferences would still allow RuralCafe to cheaply return valuable pieces of information. Specifically, for each bandwidth budget we wanted to know the grade of service a user could expect to achieve using RuralCafe.

For this analysis we first measured the sizes of the results returned by typical queries. From our search we fetched the search results page for 1000 queries to www.google.com. These search

Table 6.1: Average page sizes.

Page Type	Uncompressed	Compressed
Search Results (Full)	25.0KB	7.7KB
Search Results (Text)	23.6KB	6.9KB
Target Page (Full)	162.2KB	102.8KB
Target Page (Text)	60.2KB	18.8KB

result pages contain links to target pages and short descriptions of the target pages contents. Each search result page had an average of 13 of these links to target pages. We downloaded the target pages on each of these search result pages to get an idea of the relative sizes of pages and their contents. Along with all pages downloaded we included the embedded content. Table 6.1 summarizes the average sizes of the types of content we downloaded along with the compressed sizes. We observe that filtering out images reduces target page sizes a great deal (62.9% less) as compared to search result pages (5.6% less). Compression helps reduce search result pages and target pages dramatically in size especially text pages (70%). These results indicate that a great deal of bandwidth can be saved depending on the users search preferences.

Cost of Search: As an example, for a 100KB search result budget, RuralCafe is able to return 10 search pages with 10 results and short descriptions per page. Alternatively, the same 100KB budget could be used to return a full target page. The preferences are useful for a wide range of search budgets starting at even 100KB. Searches cost only fractions of cents in a variety of different settings using 100KB budgets over satellite in India, and smaller 10KB budgets over GPRS in Africa.

Search Response Time: To understand how well RuralCafe performs in different network conditions we performed microbenchmarks under different emulated settings. For different search result budgets we varied the bandwidth of our emulated link while leaving the network latency

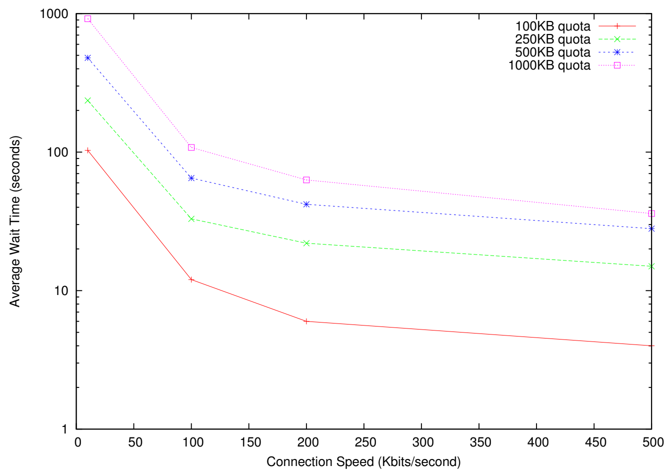


Figure 6.4: Average wait time vs connection speed.

at under 10ms. We performed 1000 search queries using the default RuralCafe richness and prefetching settings. Figure 6.4 shows the resulting average wait time depending on the connection speed. We can see that RuralCafe is capable of providing service across the spectrum of connection speeds constraints on average wait time or cost per query.

Since RuralCafe has not been optimized for speed some processing time is necessary at the proxies to service each query. The overhead is negligible compared to the time spent transferring data across the slow connection.

6.6 User Study

To evaluate the effectiveness of the various modifications on improving the search experience in high latency, low bandwidth environments, we conducted a within-subject experiment comparing two versions of our system to search using a conventional web browser (Firefox 3.0) with text-only enabled by default, which we refer to as *Conventional*. We used RuralCafe in two different configurations to explore the design space of asynchronous queueing systems for web browsing. The first RuralCafe configuration only used the local search capability (*LocalSearch*). We believed that local search alone could provide some performance benefits to users by itself. Also, this system was the most closely related to TEK [245] in terms of core features (queueing and local search). The second RuralCafe configuration enabled both local search and prefetching (*Prefetch*).

Twenty students (6 male) between 22 and 24 years of age from the same university as our preliminary study in Section 2.3 were recruited as volunteers from the student body to participate in this study. Our participants could not uniformly commit more than one and a half hours to the study because of classes and scheduling. Due to these time constraints, we formed two gender balanced groups of ten participants each wherein each participant used one system (*LocalSearch* or *Prefetch*) for 30 minutes and *Conventional* for 30 minutes. The order of the conditions was counterbalanced in each group to reduce ordering effects.

Participants were first given a short demographic questionnaire before the start of the study.

During each condition, participants were given one of two sets of five informational search tasks each. Each task was designed to require multiple searches and each task set had goals of comparable difficulty resembling both focused and open ended search tasks such as:

"What state in India has the largest population, and what is the population?"

and

"Pretend you are trying to find a cheap digital camera, find two possible cameras along with their prices."

The five tasks were given to each participant all at once before starting each condition and participants were told to complete as many tasks as possible. Task completion was self judged by the participant, the answer written down, and confirmed by the observer. Giving participants five tasks at once was intended to determine how well our mechanisms helped with multitasking.

A final exit questionnaire was given at the end of the study asking questions including their preference for the modified or the standard web browser and the three best and three worst things about the modified system they used. All participant actions on the web browser were logged during the study.

The only difference in the environment from our initial formative study was that the bandwidth for each experiment was throttled to 50Kbps for consistency. This environment represents a low bandwidth connection as opposed to a high latency or intermittent connection. The browser used for this study was Firefox version 3.0 with multiple tabs enabled. Also, the screen available for use in this study were considerably smaller (15 inches) than in the formative study (19 inches).

We did not have access to the contents of the existing cache, so to provide a realistic cache, we manually warmed a local proxy cache by iterating once through the task sets in our study. For our study we bypassed the existing proxy completely, and used our warmed cache in its place. We used a fresh copy of this warmed cache for each participant for each condition *including the baseline* for fairness. From our results the cache hit rate was approximately 19.5%, which is comparable to the actual cache hit rate of the Squid proxy cache at the university (10-25%).

6.6.1 Quantitative Results

We perform all of our analyses on the logged data obtained from our study using mixed-model analyses of variance with repeated measures because our experiment was a mixed between- and within-subjects factorial design (with participants in group one using *LocalSearch* and *Conventional* and participants in group two using *Prefetch* and *Conventional*). All of our models include Method (*LocalSearch*, *Prefetch*, *Conventional*) as a fixed effect and Participant (nested within

Group) and Task Set as random effects. Modeling Participant accounts for individual differences in performance and modeling Task Set accounts for any difference in the difficulty of the individual tasks. Note that mixed-model analyses can appropriately handle the imbalance in our data resulting from both groups using *Conventional*. We also performed post hoc pairwise comparisons using sequential Bonferroni corrections when applicable. Throughout this section, we also report least-squared means obtained from our mixed-model analyses.

Number of Tasks Completed

This metric measures overall performance. We expected task completion to increase with the *LocalSearch* and *Prefetch*. However, we found no significant difference in the *Number of Tasks Completed* using either of our asynchronous systems (2.67 using *LocalSearch* and 3.14 using *Prefetch*) and *Conventional* (2.84 tasks).

Round Trips

The conventional search process using standard web browsers proceeds as follows. First, a person issues a query to the search engine and then the server returns a search results page. This is one 'round' trip over the network. Then a person looks at the results and if he does not find the desired information he reformulates his query and requests new results incurring another round. If a page has embedded objects such as images or scripts, the browser automatically requests them from the servers resulting in another round of communication. While this process is not a problem for fast connections, when the connection is slow, each round incurs substantial idle time for the user. In the case of *Conventional*, the number of round trips is the sum of all *Search Requests* (this measures the search requests made by the user to the search engine) and *User Initiated Page Requests* (these are the requests made by the user explictly) that were not found in the cache. Images are disabled by default in *Conventional*, but scripts are often required for the

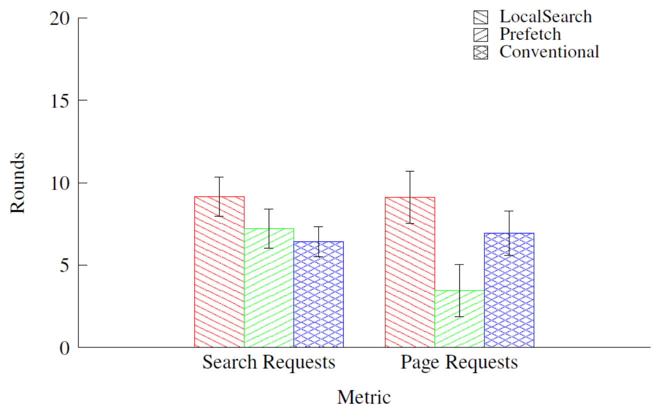


Figure 6.5: Number of *Round Trips* broken down into *Search Requests* and *User Initiated Page Requests*. Total time spent browsing is the same in all cases. Prefetch has fewer page request rounds.

complete rendering of a page. We give *Conventional* the benefit of not counting these against the number of page requests. Thus, the total number of requests is a lower bound on the true count. In the case of *LocalSearch* and *Prefetch*, the number of round trips consists of the sum of the number of remote *Search Requests* and remote *User Initiated Page Requests*.[2] We expected that the number of round trips incurred by these remote requests would decrease somewhat for *LocalSearch* and moreso for *Prefetch* as a result of users performing more local searches and browsing before issuing remote requests. Surprisingly, we found no significant effect of Method on the number of *Search Requests* (Figure 6.5: *LocalSearch* = 9.16, *Prefetch* = 7.21, *Conventional* = 6.41). We did find a significant effect of Method on the number of *User Initiated Page Requests* (*LocalSearch* = 9.12, *Prefetch* = 3.47, *Conventional* = 6.95, $F_{2,24.5} = 6.24, p \approx .006$). Post hoc pairwise comparisons show significantly lower *User Initiated Page Requests* experienced using *Prefetch* compared to *LocalSearch* ($F_{1,35.9} = 12.3, p \approx .001$) and *Conventional* ($F_{1,23.1} = 6.36, p \approx .019$). There was no significant difference between *LocalSearch* and *Con-*

[2]For these two metrics we only report results for the remote requests since they incur waiting time for the user.

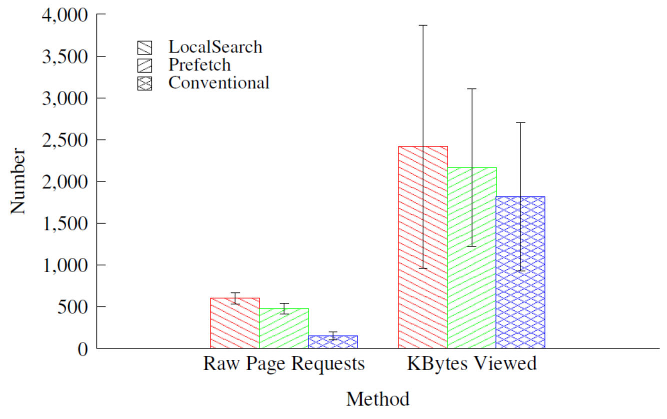


Figure 6.6: Number of *Raw Page Requests* and KBytes viewed per participant. More raw page requests and Kbytes viewed for LocalSearch and Prefetch than Conventional.

ventional.

Raw Page Requests

Raw page requests are the total number of pages requested by the user and the browser on the user's behalf (i.e. scripts, applets, etc.). This metric is meant to capture the amount of information requested and actually viewed by the user. For *Conventional*, the *Raw Page Requests* are simply all requests (both user initiated and browser initiated). For the asynchronous systems, the *Raw Page Requests* are similarly measured using only local requests (excluding UI frames). Remote requests are intentionally excluded from this metric because users may queue requests and not actually view them.

We found a significant effect of Method on the number of *Raw Page Requests* made by the browser on the user's behalf (*LocalSearch* = 601.9, *Prefetch* = 476.4, *Conventional* = 155.9, $F_{2,9.23} = 10.1, p \approx .005$). Figure 6.6 illustrates the *Raw Page Requests* along with KBytes of data actually viewed per participant for comparison. Post hoc pairwise comparisons show that the significant differences were between *LocalSearch* and *Conventional* ($F_{1,11.3} = 20.6, p \approx$ .0007), and between *Prefetch* and *Conventional* ($F_{1,11.3} = 20.6, p \approx .0007$). This is presumably beneficial to the user, as more content is delivered with the asynchronous queueing interface per unit time.

Cache Hit Rate

The cache hit rate is the percentage of *Raw Page Requests* that were found in the proxy cache in *Conventional* or in the local proxy's cache in *LocalSearch* and *Prefetch*. In terms of the *Cache Hit Rate* of the number of objects, we found a significant effect of Method (*LocalSearch* = .099 (9.9%), *Prefetch* = .277 (27.7%), *Conventional* = .195 (19.5%), $F_{2,27.1} = 3.57, p \approx .042$) with the only significant difference between *LocalSearch* and *Prefetch* ($F_{1,30.4} = 7.23, p < .011$).

Bytes Downloaded

The number of bytes downloaded is measured as the total size of objects counted in the *Round Trips* metric. As expected, we found a significant effect of Method on *Bytes Downloaded* (Bytes) (*LocalSearch* = 864,735.9, *Prefetch* = 7,809,582.8, *Conventional* = 755,836.6, $F_{2,37} = 41.2, p <$ .0001) with over 10 times more *Bytes Downloaded* by *Prefetch* compared to both *LocalSearch* and *Conventional*. The post hoc pairwise comparisons show that the signficant differences were between *Prefetch* and *Conventional* ($F_{1,37} = 71.6$, $p < .0001$), and between *Prefetch* and *LocalSearch* ($F_{1,37} = 50.8, p < .0001$).

6.6.2 Qualitative Results and Observations

Results from the exit questionnaire are summarized in Table 6.2. Of the *LocalSearch* group five people preferred the asynchronous system (50%), four preferred *Conventional* (40%), and one

Table 6.2: Exit Questionnaire Results Summary.

Statement	% Agree
Prefer LocalSearch over Conventional	50%
Prefer Prefetch over Conventional	70%
Frustrated when using LocalSearch	30%
Frustrated when using Prefetch	20%
LocalSearch more effective than Conventional	50%
Prefetch more effective than Conventional	70%
Asynchronous system is not easy to use	20%
Text only browsing is useful	80%
Local search feature is useful	60%

person had no opinion (10%); for *Prefetch* the preferences were (70%), two (20%), and one (10%) respectively. Three people using *LocalSearch* reported being frustrated using the system (30%), and two people were frustrated when using *Prefetch* (20%). Half of our participants in group one felt that *LocalSearch* was more effective than *Conventional* (50%), and seven in group two felt that *Prefetch* was more effective than *Conventional* (70%). This is interesting because we found no statistically significant increase in task completion. This may indicate a participant bias toward *LocalSearch* and *Prefetch* or simply a perception that these systems were more efficient.

The three most common "best things" about either system were in reference to multitasking, queueing, and improved speed. By far the most common response to the "worst thing" about our system was the UI. Specifically, some comments were that it was "unfamiliar" and "confusing".[3] The screens used in this study were only 15 inches, which meant that the split frame format of our UI was severely detrimental to browsing space. We designed our UI assuming that the large

[3]The UI used by the participants was an earlier version of the one presented in Section 6.2.

screen from the formative study would be available. Other negatives were that the pages were sometimes broken and queued requests could be removed, but not interrupted after they started. Interestingly, only four out of the total twenty participants reported that the asynchronous system was not easy to use (20%). Sixteen participants out of twenty said that text only browsing was useful (80%), and twelve out of twenty found local search useful (60%).

6.7 Related Work

There has been considerable work in web search and browsing in general, but in the context of emerging regions, browsing has been largely ignored in favor of research devoted to establishing Internet connectivity in the first place [129, 209, 214]. Even this first step is difficult, and only low cost solutions are more likely to scale [141, 209]. These systems allow connectivity, but in many cases the resulting connection quality is still poor and often intermittent.

Intermittent links also occur in a variety of settings outside of emerging regions. These settings include flaky wireless links [91,147], mobile nodes continuously changing access points [257], vehicular networks in urban settings [91, 199], postal networks [247]. One of the most well known systems to cope with intermittent links of longer timescales is the Delay Tolerant Networking (DTN) architecture [129]. Researchers interested in DTNs have developed several routing and addressing protocols to route packets between any pair of nodes within the network [154]. RuralCafe operates independent of the underlying routing algorithms, and our general philosophy differs from DTN systems in that the poor connection should be exposed to the user in an actionable manner.

Software solutions that reside solely in the application layer such as loband [37] (which provide compressed text-only page rendering) fail to solve the access problem in emerging regions because they do not address the condition of the underlying network. One of the earliest cross-layer systems to attempt to tackle this problem with any success has been TEK [245]. TEK allows queueing of pages for asynchronous download over simple mail transfer protocol (SMTP), and local search functionality.[4] However, it remained unclear which mechanisms benefited users to what extent and how generalizable those results were to different environments.

The web optimizations that we leverage such as caching and prefetching are extremely well studied in conventional settings and there have been many enhancements that have been proposed for low-bandwidth networks [151, 220, 256]. The work by Du [125] analyze web access traces from Cambodia to analyze the effectiveness of simple caching strategies in emerging regions. A followup work by Isaacman and Martonosi [151] show the potential for collaborative caching and prefetching in rural emerging regions. Specifically, their result shows that prefetching appropriate pages can enhance the power of local cache-based search in rural traces. These caching and prefetching strategies are synergistic with RuralCafe's local search functionality.

Alternative work in improving web access in the emerging regions attempts to increase the value of each piece of computing hardware by multiplexing these physical resources via software systems [86, 210]. These systems improve the efficiency of computer hardware at the user endpoint rather than the usage of bandwidth provided by networking hardware in the infrastructure. Still other efforts in this area are designed for specific network configurations such as peering behind individual upstream bottlenecks [226], collaborative caching behind a shared bottleneck link [151], or wireless mesh networks [149].

The potential for information access via mobile devices has recently generated substantial interest in the development and web search communities as these devices mature into web search platforms [159, 160, 237, 255]. Due to the imbalance between the size of content on the web and the capabilities of mobile devices, some efforts have opted to modify the web content itself to be

[4]TEK was designed for and studied with people who were economically deprived so the goals and results of this work are different, but much of the RuralCafe design was inspired by TEK.

more mobile friendly [9]. Mobile web standards have been designed and refined to that end, but this approach places the burden upon content developers to create and maintain separate versions of their websites.

6.8 Discussion

We found that the *Number of Tasks Completed* exhibited no significant differences across conditions. This may be the result of including some subjective tasks where completion was judged by participants themselves. Including only objective tasks may be a better method of assessing task completion ability. Since the *Number of Tasks Completed* did not yield a significant result, we look at other metrics to understand the benefits and tradeoffs offered by local search and prefetching. We also discuss some of the realities and experimental considerations that led us to our study design and limitations.

6.8.1 Local Search

We found that there were more *Search Rounds* in *LocalSearch* than in *Conventional*. If however, we look at the components of *Search Rounds*, *Unique Search Requests* and *Unique User Initiated Page Requests*, there was no significant difference for either of these between *LocalSearch* and *Conventional*. This, along with no significant difference in the *Number of Tasks Completed*, shows that although local search does not improve the overall performance, it also does not worsen it.

One interesting finding was that the activity level (*Raw Page Requests*) of *LocalSearch* increased over three times compared to *Conventional*. Also, despite this, the difference in *Bytes Downloaded* was negligible. Since the total task completion was the same, this could indicate that people were able to request and view more information at no cost.

6.8.2 Local Search with Prefetching

Link prefetching is a well known method to use the idle time to download pages that are potentially useful to the user [130, 202]. The goal of this mechanism is to reduce the wait time for the successive page requests. Prefetching is beneficial for both low bandwidth and high latency environments because the number of round trips is reduced by downloading useful pages while the user is idle.

In our study the main difference we found was a reduction in the number of over the network requests for user initiated page requests with *Prefetch* compared to *LocalSearch* and *Conventional*. Our results indicated that 70% of users preferred *Prefetch* to *Conventional*. We argue that fewer round trips is a positive result particularly for network scenarios with higher latency than our own. Given a network configuration and a fixed epoch of time, only a limited number of network requests will be satisfied; as the latency increases (and bandwidth remains constant), fewer round trips per task implies that more tasks could be completed.

We also found that even with a slow connection up to 10 times more data could be downloaded with an asychronous prefetching system than a conventional browser. This is not perfectly true in our shared network environment, but this finding supports the use of prefetching for individuals with dedicated connections who do not compete for bandwidth.

6.8.3 Realities and Considerations

Due to the nature of our experiments being conducted in the field we encountered three practical difficulties and experimental design concerns that affected our study. Conducting the study in a controlled laboratory environment was not an option as there was no separate network available in the vicinity. While we would likely be able to control for some aspects of the study, the different environment would have taken away from the ecological validity of conducting the study in a realistic setting.

First, because the university network was in actual operation, we were not given direct access to the cache by the university's system administrators. Thus, we were not allowed to copy the cache and use a copy for each experimental condition and user (which would have been ideal). Simply using the cache directly would have caused contamination. Disabling caching altogether would bias comparison between the baseline and either of our systems. Given that the hit rate of our warmed cache was within the range of the observed cache hit rate at the university, we considered this to be an acceptable tradeoff.

Second, because school was in session, our participants could not uniformly commit to more than 90 minutes to the study (30 minutes per condition plus questionnaire and interview time). Given that the phenomenon of interest was slow connections, an inherent tension existed between setting the bandwidth throttle lower to observe more effects and higher to accomodate user time constraints. We opted for the middle ground, but acknowledge that ideally, a longer running study with more tasks and lower bandwidth would yield better results.

Third, we would ideally compare idle time between the different systems, but measuring idle time directly was difficult for this study. We allowed multiple tabs in our study to allow users who preferred to multitask the freedom to do so. As a consequence, we found that stopwatch timing the screencapture of our participants was too imprecise, due to the significantly increased activity. Multiple tabs in conjunction with page download times and page rendering times make determining whether people are actually idle by automatically parsing activity logs error prone since it is unclear when pages are actually available. These are some of the practical reasons we decided to use the number of network *Round Trips* as a primary metric rather than idle time.

6.9 Chapter Summary

In this chapter we described RuralCafe, a system that is specifically designed to enable users to efficiently perform web interactions. The typical user's search pattern of iteratively refining queries is untenable under extremely poor network conditions. Traditional techniques such as compression and caching are not enough to mask the poor connection from the user. RuralCafe desynchronizes web interactions from network conditions. In RuralCafe, many tasks may be performed and queued in while offline. The actual retrieval of information over the network is only done when absolutely necessary and the network is available. Users are given actionable feedback and may decide how to use their limited network resources.

We showed that RuralCafe enables efficient asynchronous web interactions over a range of different networking environments. Our system level metrics showed that overall asynchronous browsing is on par with conventional web browsing in our environment. We also found several potential performance improvements as a result of asynchronous browsing. First, we found that queueing plus local search increased both the number of raw page requests and the bytes seen by the user per unit time. Second, we found that the inclusion of the prefetching mechanism reduced the number of round trips and increased the number of bytes downloaded per unit time. We hypothesize that in higher latency or intermittent network environments the synergy between queueing, local search, and prefetching is likely to have an even greater positive impact.

Chapter 7

CIP: A Vertical Web Access Layer for Disconnected Web

The previous chapters explored progressively worse network conditions and the systems that could be designed to address them. In this chapter, we develop a system for enabling *offline* web use to satisfy the information needs of completely disconnected communities. We describe the design, implementation, evaluation, and pilot deployment of an automated mechanism to construct *Contextual Information Portals* (CIPs). CIPs are large searchable information repositories of web pages tailored to the information needs of a target population. We combine an efficient classifier with a focused crawler to gather the web pages for the portal for any given topic. Given a set of topics of interest, our system constructs a CIP containing the most relevant pages from the web across these topics.

7.1 Motivation

In the broader scope of using ICTs to serve the underserved, the focus of much effort is on the "bottom of the pyramid". Many recent ICTD projects have focused on constructing information portals or repositories for the illiterate and focus on challenges resulting from using voice as the communication and publication medium [169, 172, 208]. Typically, after such a system is constructed, locally generated content is the main source of information available. While this information is highly context appropriate to the local population, it does not leverage the huge amount of digital content already on the Internet. In contrast, our target populations are assumed to be literate, but disconnected. While populations with computing resources and a lack of connectivity are a minority, this scenario is challenging and worth solving as a first step before considering additional conflating constraints.

Contextual Information Portals represent a powerful and new paradigm for information access in regions without sufficient information sources and networking infastructure. CIPs present a fundamentally different model of information access since they can be stored on any large storage media (e.g., hard disks, DVDs, USB Keys, or SD Cards), and shipped as a self-contained web cache to any geographic location. CIPs are complementary to existing mechanical modes of bulk digital information dissemination that makes them easily deployable [137, 154, 215, 227, 228]. CIPs may also be used to address existing connection or content scarcity by bootstrapping local web caches behind slow connections, as a stand-alone portal within a kiosk service, or as an information source for low cost digital information distribution using TV and DVDs [137]. We also plan to miniaturize CIPs for use with cheap mobile devices. Since CIPs can be easily tailored to any topics of interest on any browser-capable device with sufficient storage, we envision even small grassroots organizations or NGOs with comparatively few computing resources building their own CIPs for field use on smartphones (Third-parties could supply customized

CIPs as a business).

There are several specific areas where CIPs could be useful for different knowledge domains and scenarios:

Agriculture: eChoupal [16] is a large initiative by ITC in India that has established nearly $7,000$ kiosks with VSAT Internet connectivity in villages throughout India to directly link with farmers for procurement of agricultural produce, especially soybeans and wheat. While the current usage model of these kiosks is limited, an India-specific information portal on soybeans and wheat could be an important value-added service that eChoupal could offer its rural farmers at each kiosk. We have begun work with agricultural universities in India to develop such portals.

Medicine: Bluetrunk Libraries [74] is a massive project by WHO to ship mobile libraries of 150 healthcare books into remote regions of Africa as an educational guide for healthworkers on the field. St. Johns Medical College, one of the largest hospitals in India has expressed interest in a similar resource for medicine. The same idea can be extended to disease specific portals for important diseases such as HIV, TB, malaria, diabetes or for portals for specific specializations such as opthalmology or surgery [69].

Education: Given the volume of educational material available online, educational portals for specific topics may be automatically constructed separately or as supplementary materials to well-structured portals such as MIT OpenCourseWare [43] or educational video repositories such as Khan Academy [34].

Locality-specific web portals: Collaborative caching over delay tolerant networks [151] has been observed to improve cache hit rates dramatically due to similar interests across nearby villages. The prefetching component in these systems could be improved by learning which location specific topics to prefetch and when time varying topics should be updated (e.g. weather and news should be updated frequently).

7.2 System Design

The basic premise behind Contextual Information Portals is that the context informs the specific information needs of communities in emerging regions. More specifically, a context refers to a set of *topics* that represent the primary interests of the target users. For example, the educational syllabus of a school could represent the list of topics of interest for students and teachers within a school. We divide designing a CIP into three high-level design constraints. First, the information stored in the cache should be context appropriate. Second, the solution should be efficient in terms of relevant pages downloaded; both crawl bandwidth and disk space are limited resources. Third, the final set of pages in the CIP should not only be searchable but also *browsable* in some fashion.

A simple way for constructing a CIP is to index an existing web proxy cache from a similar school nearby. However, such a cache was not available to us. If a compatible cache did exist, it would most likely be significantly smaller and still require crawling to achieve sufficient per-topic coverage. Moreover, the pages in such a cache would not be organized across topics, and would thus require topic classification before being useful to a crawler.

Another natural way to construct CIPs would be to issue several queries about a topic to a search engine and download the search results to construct a CIP on a topic. While the simplicity of such a design is appealing, it turns out to have several flaws. First, ranking algorithms used by search engines (such as PageRank) are global optimization functions across all pages on the web. As a result, given several different topics as queries, the same pages with high PageRank keep reappearing (such as the Wikipedia pages) as the top results regardless of their rank relative to the target topic. Second, the list of pages returned by a search engine are not well-connected by hyperlinks to enable users to easily browse the contents of the portal.

The approach we take is to use a focused web crawler to crawl the web for pages that are relevant to the specified topic. The crawl is initially bootstrapped using the search page results corresponding to a fixed set of queries about the specified topic to a search engine. Blindly crawling web pages even when given a relevant starting point is problematic: pages a few links away are often irrelevant. Building a focused crawler requires us to use a good document classifier that accurately classifies pages relevant to a topic. Next, we describe our approach, classifier and focused crawler components.

Our system involves four steps:

Step 1: Defining topics. Topics may be defined many ways. One simple way of gathering sub-topics is to use manually-generated ontologies such as Wordnet [131] or machine generated ontologies such as the suggested queries returned by large Internet search engines. In this work, we assume that a set of desired topics is provided, and in our evaluation we show how a set of topics may be defined quickly. In scenarios, where user information needs do not fall into specific topics, the notion of CIPs may not be highly appropriate.

Step 2: Training a classifier. The purpose of a document classifier is to determine whether a page is relevant to a specific topic. While there has been much work on document classification in general [127, 138], personalized or context-aware classification is still an area of active research [140, 258]. We designed and implemented a custom feature extractor that is compatible with several existing classifiers and provides high classification accuracy with minimal training.

Step 3: Focused Crawling. To perform the actual crawling process, we implemented a focused crawler based on the Shark crawler [145]. The goal of a *focused crawler* is to crawl only the relevant portion of the web that relates to the topic while minimizing the waste of downloading unrelated pages [84, 206, 258]. Our crawler uses the input of the classifier and various page and link level features to reduce the crawling of irrelevant pages. We make several modifications to the Shark crawler to improve the crawl efficiency.

Step 4: Presentation. While the crawler simply outputs a large number of web pages, it is essential to organize these pages into a searchable and browsable repository. We use a combination of existing proxy cache implementations and search indexing implementations to construct a presentation layer that presents a simple search and URL interface usable from any standard browser.

Although our system is largely automated, the human is included in the loop to guide the automation. The topics are generated by people, sites that are irrelevant are blacklisted, and presentation blacklisting useless sites, and although we do not discuss it in detail, we use a simple bookmarking tool to assist instructors in building lesson plans from the pages.

7.2.1 Document Classification Overview

Document classification plays a central role in our system. It is used by the crawler both to determine whether a particular page should be added to the CIP and also to decide on the direction for further crawling. The task of our document classifier is: given an arbitrary topic and a web page, determine whether a web page is related to that topic. Document classification is a well studied problem in information retrieval [156, 176, 183], and specifically in relation to the content on the web [217, 234, 253]. Even with this previous work, classification of web pages continues to be challenging, and existing approaches either do not provide high levels of accuracy or require extensive training.

There are two main factors that make document classification difficult: (a) noisy features and (b) topic ambiguity. First, in any document classification algorithm, extracting the right set of features plays a critical role in determining the accuracy of classification. In our case, the set of training documents often contains empty or irrelevant pages – a naive classifier will attempt to use these features as part of its model, resulting in decreased overall effectiveness. Second, many broad topics are often ambiguous, making classification of documents for these topics a hard problem. For ambiguous or broad topics, the topic may have different meanings and the topic and its related terms may reappear in various contexts. To overcome these two problems, we use a feature extraction algorithm that is compatible with statistical classifiers to improve classification accuracy.

7.2.2 Feature Extraction

The main idea of our feature extraction algorithm is to use a combination to two different and potentially opposing metrics to extract textual features for a given topic: (a) *popularity* and (b) *rarity*.

Popularity of words related to a given word is commonly used across existing classifiers [217] to weigh closely related terms within a document. Given a training set of documents related to the topic, the popularity metric determines a list of popular terms that are closely related to the topic.

Rarity is a metric that identifies the list of rare terms that are closely related to the topic. To measure rarity of any given term (which need not be a single word but an n-gram), we leverage the "web 1T 5-gram Version 1" dataset from the Linguistic Data Consortium (LDC) [36] to learn the frequency of occurrence of any n-gram on the web.

Either metric by itself may not be sufficient to achieve good classification accuracy. Also, filtering the feature set using these metrics reduces the noise in the classification model when compared to using the entire text of a document for classification. An additional advantage of our approach is the feature extraction process is fast and does not require extensive training sets. The smaller set of features also significantly reduces the time to train a model. Finally, our feature extraction algorithm may be used in conjunction with many statistical classifiers such as Bayes or SVM.

Given a particular topic, we consider the top N pages from a search engine such as Google as our candidate classification training set. To detect the relative frequency of a term t, we used the Linguistic Data Consortium dataset, which provides the web frequency of $n-$grams for $n \leq 5$. We denote this as $LDC(t)$. We use the LDC dataset to discount terms from the feature set relative to their popularity. For every term t, we compute the TF-IDF [158] value of that term as:

$$tfidf(t) = tf(t) \times log(N/N(t))$$

where we approximate $N(t)$ using $LDC(t)$ and N using the maximum value of the LDC across all $n-$grams. The LDC is a commonly used dataset for estimating the inverse document frequency of a term when no good corpus exists for the specific application under consideration.

To extract popular features, we define a threshold P_{max} that represents the LDC max value below which we extract features of interest. We define a term to be *popular* relative to the topic if the following constraint is met:

$$tfidf(t) > T_{th}, LDC(t) < P_{max} \tag{7.1}$$

where T_{th} is a lower bound on the TF-IDF value for a term to be considered. P_{max} is the upper bound on the LDC count to remove extremely popular terms from consideration. Based on manual inspection across different topics, we set P_{max} roughly equal to $100,000,000$ for the LDC dataset. (In practice, we will set $P_{max} = 2^k R_{max}$ where k, R_{max} are two parameters described below)[1] The value of T_{th} was also computed as a common base value across different topics. For $N = 100$, across $15 - 20$ focused topics spanning different areas, we found $T_{th} = 4$ to be a good separation point across topics.

[1]Note that the LDC dataset that is currently publicly available is circa 2004 and may not be reflective of the currently web frequency. No later versions were released by Google.

We use a graded measure to compute *rare terms*. The basic definition of a *rare* term is based on the following constraint:

$$tfidf(t) > R_{th}, LDC(t) < R_{max} \tag{7.2}$$

Here, R_{th} is a lower bound on the TF-IDF value and is typically much smaller than T_{th} for popular terms. For a threshold of R_{th}, we set R_{max} to be the upper bound on the LDC count to restrict this set to only consider rare terms.

Using a base threshold of R_{th} and R_{max}, we set up a graded threshold for measuring rarity. We divide the LDC frequency range in a logarithmic scale between R_{max} and P_{max} and derive an appropriate threshold for every range. For every scaling of the LDC threshold by a factor 2, we scale the TF-IDF by a scaling factor β. This introduces the following secondary condition - for $1 \leq l \leq k$:

$$2^{l-1} R_{max} \leq LDC(t) < 2^{l} R_{max} \tag{7.3}$$

and

$$tfidf(t) > \beta^{l} R_{th} \tag{7.4}$$

This represents a graded way of measuring rare terms across each range within the LDC frequency spectrum. In other words, we defined a base low value of $R_{max} = 1000$ as the smallest value under consideration and divided the LDC scale based on a logarithmic scale; we considered exponentially scaled up versions of R_{max} such as $2R_{max}$, $4R_{max}, \ldots 2^{k} R_{max}$. To extend this as a continuous measure between rarity and popularity and combine the two spectrums together, we set $P_{max} = 2^{k} R_{max}$. Similarly, $T_{th} = \beta^{k} R_{th}$. A specific choice of values appropriate for the 2004 LDC data set are: $R_{max} = 1000$, $k = 17$ and $P_{max} = 131,072,000$. Hence, our algorithm sets up a basic lower bound threshold on rarity based on R_{max} and a upper bound threshold based on P_{max} and uses a graded measure to extract features based on the terms with an LDC frequency in the range of $[R_{max}..P_{max}]$. Very rare terms with a frequency lower than R_{max} are selected as topic specific features if their TF-IDF score is greater than R_{th}.

7.2.3 Classifier

We tested our feature extraction algorithm in conjunction with two standard classifiers for our system, Support Vector Machines (SVM) and Naive Bayes. Both of these have been shown to be effective for text classification [155, 176]. For our system, we used the bow classification toolkit [10], which provides both Naive Bayes and SVM implementations. Classification was evaluated on the reduced feature set produced by the rare/popular method described above. We show later that our feature extraction algorithm improves the classification accuracy of both the classifiers.

7.2.4 Focused Crawler

Focused crawling is a fairly well explored topic. Our focused crawler is based on Shark [145], a heuristic crawler that orders the nodes to be visited by sorting them according to their similarity to their ancestors in a priority queue. The key difference between a focused crawler and a standard crawler is that a focused crawler uses a ranking function to determine which outgoing link to traverse next with the goal of only crawling relevant pages corresponding to a topic.

Designing a good ranking function to decide the set of outgoing links to follow to direct the focused crawl is a challenging task and many heuristics have been proposed. The general methodology for designing a ranking function is to estimate a probability value for every outgoing link, that it would be relevant for a topic or not. We experimented with a combination of different parameters many of which are used by the Shark crawler: (a) number of crawled pages pointing to an out-going link; (b) the relevance of the parent page(s) to the topic as output by the classifier; (c) cumulative relevance of the parent page as output by the estimation function; (d) fraction of relevant pages crawled from the parent page; (e) relevance of anchor text around the link.

We made several optimizations to the Shark crawler to enhance the performance of the focused crawler for constructing CIPs. First, the basic Shark crawler uses a Naive Bayes classifier to estimate relevance of pages and the ranking function; our feature extraction algorithm coupled with the SVM classifier is what we use instead. Second, Shark can be bootstrapped with an initial set of A *authoritative documents or authorities*. In our case, the initial set was chosen based on the search results from a large search engine; since we leverage Google based results that uses the underlying web graph, we believe our initial set to have good hubs and authorities for a topic.

Third, we added a heuristic technique called *tunneling* [93] that allows a number of irrelevant pages to be traversed to reach a relevant page. This is important because, pages relevant to a topic are not necessarily connected in the web graph. In our implementation, we set the tunneling depth to $D = 3$.

Fourth, we also made several changes to the Shark ranking function. We run the classifier on the initial set of authoritative pages to calculate a relevance probability p for each page and insert the page in a priority queue with probability p. This is particularly important for Google search results since we found that a non-trivial fraction of search result pages on a focused topic may not even contain the keyword. We introduced a scoring function where the inherited score of a child node, c, of a relevant current node depends on, the probability p assigned by c as the relevance of the current node.

Finally, we use the anchor text around the hyper-link by extracting text from a fixed number (currently set to 2 words) of surrounding markup elements. We passed the anchor text through the classifier and used the classification probability in the ranking function.

7.2.5 Presentation Layer

The use of a CIP is similar to that of a normal web browsing session except that a CIP is completely offline. Similar to a search engine, CIP uses a simple presentation layer of a search interface and a list of high-level topics covered by the portal. Once we obtain the list of web pages from a focused crawler for a given topic, we use Lucene [39] to index the documents for a given topic. We also provide a simple mechanism for a user to search the list of topics within the portal. We also save the HTTP response headers, URL, and page title in the index for search and presentation. To make navigation easy, before presenting any webpage to the user, we also highlight the links present in the CIP to enable a user to navigate across pages within the cache.

Apart from a search interface, a user accesses a CIP using standard URLs. From a web browsing perspective, a CIP resembles a simple proxy cache in that if a page referred by a URL is present in the CIP, the page is directly accessible by its URL.

7.3 Implementation

We implemented our crawler in approximately 5000 lines of multi-threaded Python code. Our crawler is not a parallel crawler because at each iteration the main crawling algorithm downloads a page and computes the most promising next page to download. Threads are only used to parallelize tasks within this serial processing framework. We allocate a thread pool of up to 50 worker threads per process. Each thread may be given one of three tasks: *seedTask*, *embeddedTask*, *frontierTask*. The main thread performs the crawl and assigns tasks to the worker threads as necessary, and waits for the threads to complete or time out before continuing. The classifier runs in its own thread and called to classify documents by each worker thread. We use Lucene [39] to index the documents.

During execution, the main thread begins by requesting the URLs of A authorities and assigns each URL to a seedTask thread. Each seedTask thread downloads an authority page and adds it to the queue of “nodes”. The main thread then begins crawling until either the queue is empty or the quota of pages to download is reached. For each node in the queue, the main thread classifies and caches the page if it is relevant. If the page is relevant, the main thread finds any embedded object references and assigns embeddedTasks to download the objects. We have implemented parsers for detecting embedded objects referenced by HTML 2.0 to 5.0 and CSS object references. Since our crawler does not execute Javascript, dynamic objects are not fetched. Finally, the main thread gathers the outlinks on the page if they exist and ranks them using the classifier again based on the anchor text and surrounding text. After the quota is reached, the main thread assigns frontierTasks to download the text-only pages in the frontier.

To demonstrate the portability of CIPs we integrated our repository with several user interfaces. First, we used Carrot2 [12], an off-the-shelf open source document clustering and interaction interface. We found that while the UI presented by Carrot2 was visually appealing, the automatic clustering algorithm did not always compose topics that facilitate the process of finding information targets. It was definitely useful for getting a sense of the general ideas in the CIP. We also integrated our repository and index with the RuralCafe user interface described in the previous chapter. The benefit of this is that if there is some network available, RuralCafe could dynamically update its contents and slowly improve the CIP on the fly. Carrot2 uses its own indexing mechanism that required slight modification to the crawler. Other UIs and presentation formats may require modification to the document and indexing mechanism, but these two adaptations required the addition of only a few lines of code.

Subject	# of Topics	Example Topics
accounting	408	ledger, business transactions, budget
agriculture	489	farm layout, livestock production, potassium
arabic	39	adjectives, particles, inflexion declension
art and design	433	insects, finishing, cubism picasso braque
biological science	647	fish, life cycles, cell physiology
chemistry	778	scientific objectives, salts, forces
christian religious education	475	apostleship, african religious heritage, luke
commerce	444	supermarkets, supply, promissory notes
computer studies	658	printing, internet software, fibre-optic cables

Table 7.1: Example subjects with topic counts and example topics.

7.4 Microbenchmarks

To evaluate our system we constructed several CIPs based on the syllabus of a Kenyan secondary school. For each subject (Table 7.1), the syllabus contained requirements and course descriptions. We extracted the topics from the syllabus by unbinding and scanning the relevant pages of the hardcopy, OCRing it into a digital copy, cleaning up the topics automatically using several noise filters to remove garbage symbols and a simple stopword filter.[2] For the purposes of evaluating our system we did not further clean or manually rewrite any of the topics. If a topic was poorly defined, it is easily identified during the feature extraction process and may be rewritten manually. A sample of the topics we extracted is shown in Table 7.1. Using this set of topics, we evaluated our system first using microbenchmarks of the classifier and crawler. We then summarize our experience with a CIP generated for a preliminary study, our improvements following that study, and demonstrate the ease of deployability of our CIPs by setting up

[2]These steps could be eliminated if we had a softcopy.

a larger-scale pilot at 5 schools in Nairobi.

7.4.1 Classification

First we perform several benchmarks on our classifier using these authentic topics and complement them with several experiments using the WebKB dataset to briefly compare the performance of our feature extraction algorithm against previous work.

Feature Extraction

To evaluate our feature extraction algorithm for a given topic, we generate a training set of candidate documents based on search engine queries for the topic and extraction of the text from the resulting pages. Filtering is applied to this initial set of documents to remove uninformative documents. The remaining documents are randomly partitioned to generate training and test data sets. In addition a negative test set is generated by extracting 10000 random documents from the English wikipedia corpus.

We verify that the number of features kept by the rare and popular filters is very small relative to the original feature counts. The number of features kept by the filtering process is shown in Figure 7.1.

Classification Results

We tested two classifiers on our corpus of topics, a Naive Bayes classifier and an SVM learner; we used the bow [187] toolkit to construct and test our classifiers. The models used for the classifiers were generated using the text of the documents as a unigram bag-of-words model. The classifiers were separately trained and tested using a set of words corresponding to the union of the rarity and popularity filters.

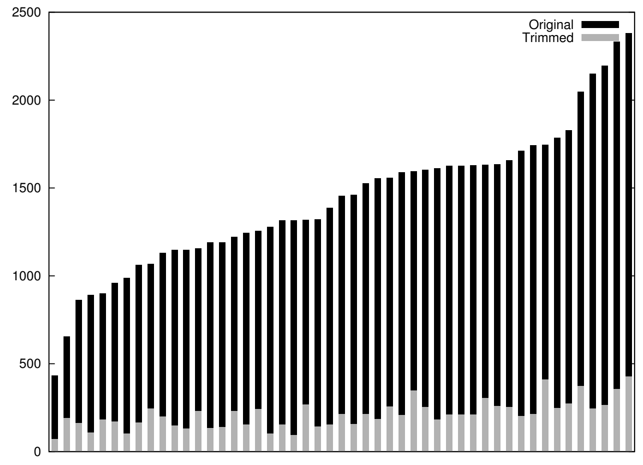


Figure 7.1: Total and filtered terms, per topic.

Our classifiers were run with their default settings. For libbow, this generates a learner based on a unigram word model. The Naive Bayes learner is smoothed by assuming a Dirichlet prior for zero valued features. The SVM learner uses a linear kernel with linear weighting on term frequencies.

The results of evaluating our topics were surprising - both the SVM and Bayesian classifier had very high precision the full range of subjects.

For this particular classification task, we found the effectiveness of the Naive Bayes classifiers to be significantly better then our SVM learner when training against the full feature set; the reverse occurs when training against the restricted word set. The effect of filtering terms from the feature set dramatically altered the behavior for our classifiers, in differing manners.

When working on the filtered set, our Naive Bayes classifier lost a small amount of recall, and showed better precision when rejecting documents of the negative test set. The classifier exhibited perfect precision in rejecting random documents on 3 of the topics. The most dramatic change was within our math topic set. We suspect the reason for the loss of recall to be related to the distribution of words within the math topic: the number of related but uncommon words for that topic is significantly larger then for the other sets.

When run against the filtered set, the SVM learner showed a uniform improvement in recall (acceptance related documents) for all categories and trivial losses in precision (rejecting unrelated documents). The SVM learner operating against the filtered feature sets outperformed all of our other classification attempts by a significant margin.

Finally, we observed qualitatively that the time to train the SVM was drastically reduced when running on the filtered features.

WebKB Evaluation

As a base of comparison with related work we evaluated our techniques against the WebKB dataset, though we did not expect to achieve groundbreaking results in this area. The resulting performance was better then anticipated. We followed the procedure used in [196], and focused on the joint classification of the reduced set of 4 groups: course, faculty, project and student classes. Once again, we saw differing behavior in our classifiers. The aggregate results between SVM and Naive Bayes are displayed in Table 7.2. For this document set, our naive Bayes learner on the trimmed features outperformed the other classifiers - achieving an aggregate accuracy of 90.7%. Without feature filtering, the Bayes learner achieved only 81.7%. The SVM learners exhibited the reverse behavior - the learner operating on the trimmed set exhibited significantly decreased accuracy.

Upon investigation, we found that our thresholds for this data set had been too aggressive - only 7 words were being kept for the "student" class via the popular filter. The lack of features to work with significantly handicaps the SVM learner. The naive Bayes learner, however, extracts most of the weighting for classification from exactly these rare terms, and therefore does not

SVM (Original)	89.8%
SVM (Filtered)	80.2%
Naive Bayes (Original)	81.7%
Naive Bayes (Filtered)	90.7%

Table 7.2: WebKB classification accuracy.

exhibit the same degradation; the removal of spurious terms prevents the classifier from over-emphasizing them.

We note that the results for the filtered Bayesian learner are roughly on par with the best techniques we are aware of. This is somewhat surprising given the relative simplicity of our approach.

7.4.2 Crawling

Until now we have only been measuring accuracy of our classifier using our training set as the basis for comparison. To get a sense of how well our crawler and classifier worked for realistic topics we ran our crawler across each topic in the syllabus. We ran our crawler on cluster of 12 Duo and Quad core machines each with between 2 and 8 GB of RAM on the 658 computer studies subject topics. Across our cluster the crawlers requested $12,318,752$ files, of which, there were $2,284,722$ unique files from $53,685$ domains.

Harvest Rate

The harvest rate for topics in our corpus in Figure 7.2. We found that the harvest rate for 82% of topics in computer studies remained above 90% efficiency. This value is not directly comparable to other focused crawlers in the literature, but it does give a sense of the absolute efficiency of

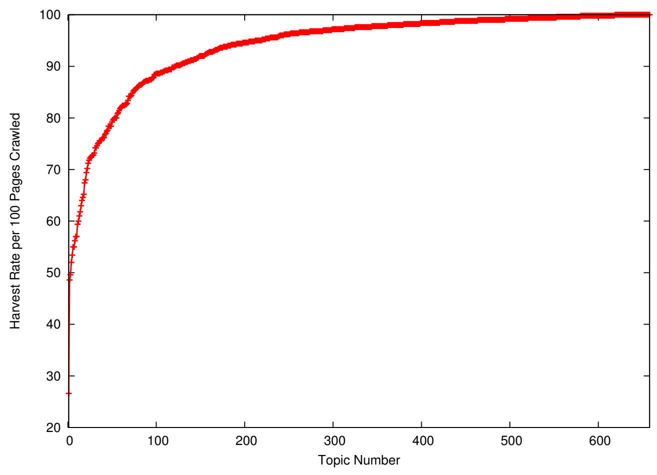


Figure 7.2: Harvest rate per 100 pages crawled.

our system. Also, while this high crawl rate that indicates that our overall system is performing efficiently, the existence of topics that our system struggles with reinforces the need for an efficient focused crawler and feature reduction algorithm.

Crawling Speed and Scalability

Our crawler was not optimized for speed. Using a 1.8Ghz Duo Core Intel PC with 2 GB of RAM and a 2 Mbps broadband connection the crawl rate was on average approximately 1 hour per 500 page topic including time to download the frontier that is on the order of 20 - 40 thousand pages. With our small cluster of 12 machines we were able to crawl 658 computer studies topics totaling 103 GB in approximately 2.5 days. The execution profile from running our crawler was dominated by server latency and connection setup overhead. This is to some extent inevitable due to the nature of the Internet, but could be optimized by simply reducing the connection timeout and parallelizing further. The crawler speed was sufficient for our purpose.

Crawling Accuracy

While the classifier and crawler both report high accuracy and harvest rates, the final metric for the crawled results is the usefulness of the pages as judged by a human. We went through the results of each subject manually and sampled search results for a few of the topics within each subject. We found that the quality of the pages in the CIPs generated for each topic varied dramatically between subjects despite the high precision for the classifier microbenchmarks. In particular, for topics within the some subjects, e.g. music, christian religious education, german, and arabic the results were poor because the syllabi themselves lacked clear and concise topic descriptions. To address this problem, the syllabi themselves would need to be corrected such that the topics would at least result in relevant results after being queried to a search engine. We also observed that for some topics of subjects such as metalwork and drawing and design, the terminology was too vague to capture the intended meaning (e.g. "riveting", "oil cans" are ambiguous by themselves as they could have a large number of results irrelevant to metalwork). Including the subject in the query along with the topic may fix this issue when training and seeding the crawler. We found that for the topics of subjects relating to math, science, and engineering our crawler performed well; while we expect to address these crawler issues in the future, for our main studies we used the topics for mathematics and computer studies.

7.5 Deployment

Prior to our main deployment we conducted a preliminary study with an initial prototype. We constructed a CIP for mathematics for 8th standard from an after school program in Kalpakkam, India. We summarize the findings from that study, describe several system and presentation level changes we made based on that experience, and then discuss our pilot at 5 secondary schools near Nairobi, Kenya.

7.5.1 Preliminary Study

For the after school program in Kalpakkam, India we worked with one teacher and 9 of her students. The CIP was constructed using the titles of 17 chapters of grade 8 Mathematics curriculum of the Central Board of Secondary Education of India (e.g. “square roots”, “volume and surface area”, “introduction to statistics”, etc.). The portal was configured to download 500 complete web pages for each of the topics, and the set of pages downloaded was approximately 5GB in total (compressed). The total time for gathering the appropriate pages was approximately 8 hours on a single desktop machine in the U.S. with a 2Mbps connection.

We setup the portal on a computer that was already in the classroom. We then demonstrated the use of both the portal and a simple bookmarking tool for constructing lesson plans to the teacher. Software setup and training took roughly 30 minutes and 1 hour respectively. The lesson plans, experiments, and usage of the portal were designed, executed, and decided by the teacher herself without any direction from us. The teacher was given complete freedom on how to use the portal. She used the portal to teach the 8th standard material to her students who were in 6th and 7th standard. We observed her use of the portal over the course of one week, and at the end of the week we interviewed the teacher and gave the students questionnaires relating to their experiences.

All of our participants reported positive overall experience with the portal. The students felt that the portal allowed them to quickly access information and that there was more information available than usual. The teacher was also able to easily find information in the offline portal appropriate to her course topics and quickly construct a lesson plan. The portal itself was constructed quickly, and cost almost nothing to deploy in our setting. Finally, in terms of sustainability and ease of deployment, the installation and training were also quick and painless.

7.5.2 System Design Iteration

Despite both the high accuracy performance and overall positive feedback from the participants in our preliminary study, we observed several areas that could be improved. We discuss these changes at a high level as they are relatively simple.

Paywalled Sites: First, sites behind paywalls and sites selling educational products would frequently contain pages that were accepted by our classifier. This problem occurred across roughly half of the topics. While the classifier itself performed well after training on the training set, the authority pages returned by google for training the classifier were not always useful for our purposes. As a result, the pages eventually included in the CIP would sometimes contain pages with many relevant keywords, but no actual educational content. Thousands of pages from sites such as “www.tutorvista.com” and “www.sciencedirect.com” were retrieved and usable. It was somewhat surprising that pages containing advertisements were not nearly as prevalent. While our bookmarking tool was useful for mitigating this problem during a teaching session, we would ideally remove such pages completely from the CIP.

Dynamic Pages: Second, pages that were highly dependent on dynamic content were displayed poorly or incorrectly. The resulting page presented to the user would be essentially unusable. This problem is similar to previous issue with paywalled sites. While the bookmarking tool helped, dynamic pages that failed to render properly were also a waste of resources and time. For both the paywalled sites and dynamic pages we implemented a simple blacklist filter in both the crawler and the presentation software. At the crawler end, the blacklist prevented crawl time wasted on unwanted sites. At the school itself, the blacklist was left exposed to the administrator (i.e. teacher) so he/she could block useless or unwanted pages on the fly. The blacklists could easily be merged into the global blacklist at the crawler for content updates.

Dead Links: Third, due to the offline nature of the CIP and the number of pages included, it is inevitable that beyond one or two pages browsed by a user links leading to pages that are not in the CIP. In our initial prototype we indexed all pages downloaded by the crawler. We found that the text-only pages downloaded for frontier pages usually contained mostly dead links. We removed these pages from the index for our larger pilot. In addition, we implemented a Firefox browser extension to highlight links that are found in the cache so users could avoid clicking on links.

7.5.3 Large Scale Pilot

For our large scale pilot we worked with a university in Nairobi that was already involved in weekend teaching activities at 5 schools near the city (Figure 7.3). The schools were all within 75 km of Nairobi, though the distance is slightly deceptive due to the poor quality of some roads. We were requested by the teachers at the university to construct a CIP for "computer studies", which is why much of our evaluation thus far has focused on that subject.

Each school has varying computer lab resources and some required local area networking hardware and setup. Table 7.3 summarizes the school hardware setup and any additional resources we purchased to complete the CIP setup. As the table indicates, the hardware itself is typically sufficient for a full-fledged computer lab.[3] Otherwise, the schools machines were either donated or purchased recently by the Ministry of Education in Kenya. Also, the actual maintenance of all labs was not always perfect.[4] The overall additional cost for physical resources was marginal, ranging between between 50 and 150 US dollars. We visited each of the schools twice, once to assess the hardware situation and a second time to setup the required hardware and CIP. Setup time at the schools took between 30 minutes and 3 hours depending on the level

[3]School 4 had apparently lost several machines due to corruption by the previous principal.

[4]School 2 in particular was poorly maintained, 6 brand new machines in the middle of the room were plugged in, but not networked. Several other machines were not in working order due to minor issues such as broken or missing network cables/mice/monitors.

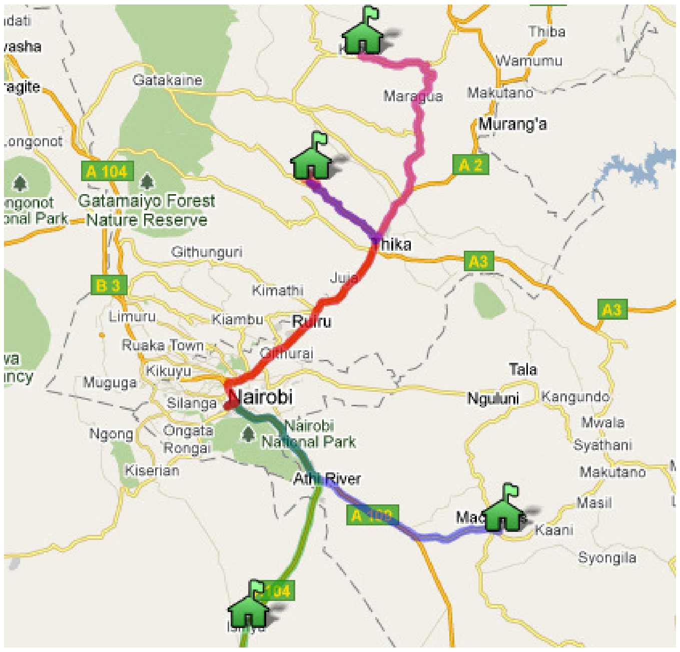


Figure 7.3: Map of the 5 participating schools of our large scale pilot.

of maintenance and people working to setup the network.

The university program involved $4 - 5$ undergraduate computer science students traveling to each school nearly every Saturday during the school year and teaching students subjects in computer studies. We decided early on to leverage this existing program to bootstrap the use of the CIPs. The students and their program manager have agreed to use the CIP for the next several months as the primary teaching resource for all teaching materials for computer studies to the secondary school students. Lesson plans and presentations will be constructed using the CIP rather than the web directly. The students themselves are qualified to teach, maintain, and upgrade the software for the CIP and resulting portals. The initial student and teacher responses at both the university and the schools have been extremely positive.[5] One teacher remarked: “Its

[5]Deployment at School 3 has been delayed due to internal conflict between the instructor and the principal. This is still in the process of being resolved.

School #	Existing Hardware	Additional Hardware
School 1	20 computers, 1 switch, 1 server	One external hard disk
School 2	23 computers, 1 switch	One external hard disk, additional network cables
School 3	20 computers, 1 switch, 1 server	One external hard disk
School 4	6 computers	One external hard disk, one 24-port switch, network cables
School 5	28 computers, 1 switch, 1 server	One external hard disk

Table 7.3: Existing school hardware and additional requirements for pilot.

been great! The students love it, I can't keep them away." and "The other teachers would like to use it for their subjects."

The software is being open sourced, and the students are encouraged to contribute by coming up with projects based on or around the CIP code base.

7.6 Related Work

The application of focused crawlers to particular topics in an effort to build digital libraries and web caches has been considered before [93, 128, 218], but the application of these automatic techniques to the context of development has not been evaluated. Vertical search portals use similar focused crawling techniques to index a specific domain on the web, and the underlying crawling and classification technology is well studied.

A wide variety of document classifiers have been developed: Decision Tree [138], Support Vector Machines [127], taxonomic classifiers [105], and many others [258]. The two techniques appearing most often for web page classification are naive Bayes [176] and SVM learners [156]. In recent years, other techniques have been proposed and used, with some success [196].

Prior work for web page classification largely focuses on feature extraction and selection. In addition to the page text, additional components of pages and their relationships have been integrated to improve classification including HTML tags [253] and the geometry of the rendered page [234].

A broad class of features for documents comes from anchors. Anchor-text from links pointing into a page can be associated with the page itself [104]. In addition, URLs and text from other pages may be accumulated into the candidate feature set. Some attempts to assign additional feature information eschew the traditional association of relatedness via anchors, and instead attempt to group together pages on the basis of their relative positions in a site tree or web graph (sibling-relationships). Still other approaches view the labeling of pages as a optimization problem on the graph of pages formed by anchors, and attempt to optimize labeling on the graph [88].

Other work in the area has been on URL-only classifiers; these ask the question of whether is it feasible to classify a page knowing only the URL (and possibly some link structure) [161]. This is of particular interest for systems such as web-crawlers and focused crawlers, where there is a desire to classify a page as desirable *before* retrieving it. Even with this restriction on available information, classification precision above 60% has been achieved on the WebKB dataset [92].

There is extensive of literature on web crawling algorithms [94, 167, 195]. Focused crawling was defined and formalized by [105], which introduced taxonomic classification or text analysis to determine document relevance and distillation or link analysis to identify authoritative sources of relevant documents. Shark is one of the earlier focused crawlers, and more sophisticated variations exist [206, 213]. Also, extensive follow up work has compared focused crawling against a variety of other crawling techniques across a variety of domains [116, 121, 189]. It has been shown in recent work [84] that uses a latent semantic indexing (LSI) classifier, which combines link analysis with text content, that a variant of the simple Shark-search can be surprisingly efficient when compared to more sophisticated and expensive techniques such as LSI and PageRank (PR).

7.7 Chapter Summary

In this chapter, we designed and implemented a complete system for automatically constructing Contextual Information Portals. We used a combination of information retrieval techniques to collect web pages for a CIP using relatively limited computing resources. We incorporated this into a complete solution and demonstrated its effectiveness for real world subject domains. We also integrated our CIP with an existing user interface to provide a portable offline digital library for communities without libraries or access to other information sources. After conducting a preliminary deployment and incorporating participant feedback and improvements into our overall system, we deployed a large-scale pilot in 5 schools near Nairobi, Kenya. Our system requires only the addition of a hard disk, and as a result both the “capital expenditure” and on-going “operating costs” are low. In the future we plan to thoroughly assess the impact of CIPs on education in our pilot, and we hope to extend CIPs to more schools and other environments.

Chapter 8

A Vertical Caching Layer for Low Bandwidth Networks

As we have seen from Chapters 5 and 6, in highly intermittent or low bandwidth environments one promising direction for improving the web experience is to improve caching and prefetching. In this chapter, we introduce a *vertical caching* layer to organize the cache around *topics* rather than caching based on URLs. In URL-based caching, unless there is an exact URL match to a cached content, a new page must be downloaded. However, in practice the same information may reside in the cache aliased under a different URL or similar equivalent content that may satisfy a user's request may be available in the cache. We implement one instantiation of this vertical caching concept and show that it is possible to quickly identify recurring topics of interest, organize a cache based on these topics, and prefetch using topics. To support vertical caching and prefetching we introduce a new cost metric for value of objects, pages, and topics based on the time spent downloading that is advantageous when the bandwidth to download time relationship is inconsistent.

8.1 Design

One of the key challenges in vertical caching is *identifying topics* within a cache. Once the topics are identified, then the vertical caching layer can organize the cache contents based on topics, make the cache topic-wise search-able and navigatable and expose the topics to the end-user. From a cache resource management perspective, vertical caching is similar to conventional caching mechanisms with URLs being replaced with topics. Consequently, we redefine the notion of a cache hit later in this chapter. Vertical caching may either be independently applied or integrated as a layer above conventional caching. In the latter case, traditional caching policies would continue to apply within the pages belonging to each topic.

8.1.1 Identifying Topics

Our system attempts to automatically identify topics from three sources of information. These sources are all based on information in the request log of the cache: URL patterns, search queries, and content patterns. Users are also able to manually specify topics that are then merged with the set of automatically extracted topics.

A topic is simply a set of related pages relevant to an interest of users. We purposely leave the general definition of a topic to be broad. In our system we define topics to be a description of a preference for a particular set of web pages. Each topic consists of: (a) description of the topic, (b) a set of pages belonging to a topic, (c) A classifier for deciding whether a page belongs to the topic, and (d) cost or value of the topic. The description of the topic is simply a label used for navigating across topics in the cache or requested to the prefetcher and used for finding more documents belonging to the topic. The description may be automatically extracted from the pages belonging to the topic or manually defined by the user. A topic description may be a URL pattern (e.g. a.com/b/*) or a set of correlated terms (e.g. search terms, or terms appearing in a set of documents). The set of pages that belong to a topic are also constructed automatically. These pages are used by the cache to display pages belonging to a topic for navigation, and by the prefetcher as seed documents for prefetching. The classifier is trained on the set of pages in the topic and used to classify new pages requested by users or by the prefetcher to download additional pages for a topic. The cost is used by the cache to evict the most useless topics and by the prefetcher to determine pages that are underrepresented in the cache based on their value.

Cost Function: A naive measure of value or cost for an object in the cache might be the number of times it was requested or the size of the object. In a traditional caching scenario, this reflects the optimal behavior: smaller objects are download faster and therefore the time spent transferring on a cache miss would be minimized. However, for challenged networks, object size is not a consistently accurate measure of the time spent transferring. Instead, what we wish to minimize is *the amount of time spent downloading*. This cost function is novel (in the context of web-caching) for determining which pages are most valuable to cache. This cost function explicitly takes into account the reality of the web browsing experience for users in high-latency environments where the cost of fetching N bytes may vary radically vary. In the degenerate case, where the cost of fetching N bytes is constant, the cost function is simply proportional to object size.

Feature Reduction: Corresponding to any topic, we use standard feature extraction algorithms to have a small set of features that are closely related to the topic. This feature set is used by a document classifier to quickly determine if any page corresponds to existing pages within a topic. We use the feature extraction and reduction algorithm developed in the previous chapter that yields high classification accuracy levels to determine the reduced feature set.

8.1.2 Domain Topics

This portion of the system examines the URL patterns that occur in an access log to identify domains and sub-domains appearing frequently enough in the cache such that pre-fetching additional pages has a high probability of producing a significant number of hits against future requests. The topic extraction engine groups web page requests into a hierarchal tree of buckets. At the top are the second level domains. Branching out from these are the sub-domains, and sub-directories within these domains. Associated with each bucket is a cost C_b. Where C_b is the aggregate user time required to fetch content belonging solely to this sub-domain. A domain topic is considered "identified" once a threshold $T = 3$ of requests are made that belong to the topic.

An example of how cost trees work with domain topics is shown in Figure 8.1 (a). After a domain cost tree has been constructed, the cost mappings are used by the cache for eviction and for resource allocation while prefetching. Leaf nodes without parse-able URLs are stored for the cost tree, but not used for prefetching seeds.

8.1.3 Query Topics

The query analysis method of topic identification consists of identifying and extracting 'query-like' phrases from the user requests to find a smaller set of more general topics. There are many ways to extract useful query topics. The simple method we use is to split requests into search sessions based on query overlap and time of request. On a per-client basis, if a query is made that overlaps with a previous query made within the last time threshold $S = 120$ seconds then the query is considered as part of the same search session. Otherwise, a new search session is created. All requests made while a search session is active belong to the current search session. The union of each query terms of each search session then form the query topic. An example of

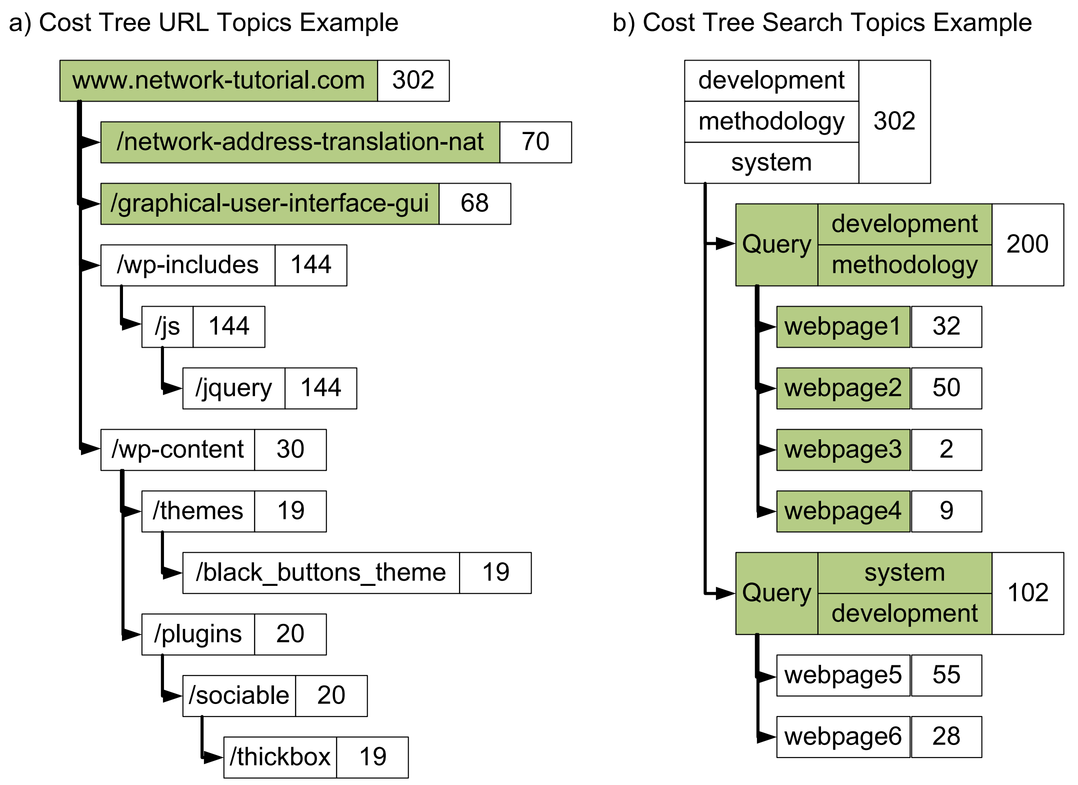


Figure 8.1: Example cost trees for (a) domain topics and (b) query topics. Costs are hierarchical and cumulative. Highlighted URLs and search results indicate seed pages.

how cost trees work with query topics is shown in Figure 8.1 (b).

8.1.4 Content Topics

Content analysis is performed using a combination of the techniques used for URL and *Query Analysis* on the text documents in the cache. As with query analysis, the desired output is a set of topic descriptions that conveys the areas of interest that users spent the most time on. However, given the time and resource constraints of the cache system, performing a clustering calculation on the full set of document content is impractical. This restriction implies a need to drastically reduce the set of documents considered and the feature set used. To help prune the search space, we reuse the domain tree constructed for domain topics to assign the relative importance of documents according to their cost. The documents are then selected highest cost first until a threshold of D documents is reached. The set of candidate document descriptions are then filtered and clustered using the feature extractor, and a final set of mappings is produced for each topic cluster. The seed URLs are the members of each topic already in the cache.

User Patterns

In addition to the methods for automatically extracting topic described above, our cache design easily accommodates manually specified topics by users. Our system can then derive meta-data for user-defined topics based on either the contents already in the cache or from the web.

8.2 Cache Management

The cache component is responsible for maintaining the pages in the cache, an index of the pages, providing an interface for searching through the cache, and for extraction of topics. During periods of synchronization, the topics are transmitted to the prefetcher for prefetching. The prefetcher is responsible for crawling of web pages for each requested topic until the amount of content per topic matches the value of the topic. The prefetcher maintains a duplicate copy of the cache and runs the same topic identification algorithm as the cache component using the cache hits received by the cache component. From these topics, the prefetcher compiles and returns the delta of the most relevant pages for each topic to the cache. As user requests arrive, they are processed by our topic identification algorithms in a background thread that identifies and updates topics in 60 second batches. The vertical caching subsystem maintains separate dictionaries for domain topics, query topics, and content topics. The topics that each page has been identified as belonging to, along with the page itself, are stored in a search-able index.

8.2.1 Updating

Updating the cache is straightforward. In the low bandwidth scenario user requests are automatically integrated into the cache. Otherwise, topics are updated if the network is available and there are no outstanding web requests. In the case of a mechanical backhaul, there is typically a mass storage device used to physically courier data, or a mobile computer or device travels between the remote location and a place with Internet connectivity. For these mechanical backhaul scenarios, a "sync" is required when the mass storage media is present. Any new pages are copied from the storage device onto the proxy and integrated with the cache. Any new user requests are copied onto the storage device. Similarly, at the connected end the topic updates are performed using the new user requests.

Our system uses prefetching to supply additional data to the cache during periods of idle bandwidth or over a mechanical backhaul. To maximize the utility of the documents being prefetched, the prefetcher needs the ability to identify documents that are of interest to users. Documents are interesting if they belong to the topics that users are interested in and have a high cost.

8.2.2 Vertical Prefetching

Vertical caching may be directly extended to prefetching to retrieve relevant pages corresponding to a topic. Our prefetching algorithm uses the set of pages within the cache belonging to the topic as a seed set and uses a *focused crawler* to prefetch appropriate pages corresponding to the topic. The prefetching is largely used for bulk-updates using the mechanical backhaul links. For incremental updates, we restrict our focused crawler to a maximum depth of 2. Pages are prefetched per topic proportional to the value of the topic across all topics until available space on the portable media is exhausted. Space on the media is reserved for the manifest that contains a delta of the filenames from the prefetcher or a list of the page accesses from the cache. Upon receipt of a topic pattern, the prefetcher begins a crawl for pages related to the topic using the seed pages and classifier. Once the size quota is met, the crawler waits for the next topic.

8.3 Organization and Presentation

Recall that in offline mode all requests by the clients are served directly by the proxy that exposes the cache and its topics. The proxy exposes the topics to the user via two mechanisms: offline search and "faceted browsing". Faceted browsing is a way to browse a set of documents based on metadata [243]. Once topics are identified with their constituent pages, the pages are indexed using the topic features as the metadata to enable faceted browsing over topics. Furthermore, to aid navigation of pages, we rewrite URLs within pages to highlight locally cached contents in a page. Cached documents are modified to indicate that they are local and potentially stale. In offline mode, all search requests are intercepted by the cache and the appropriate search response is provided depending on the cache contents and the topics. URL requests that cannot be serviced by the cache are given a plain 404 response along with links to potentially similar pages if any exists. We have considered several alternative designs for further exposing the cache for direct interaction, but a thorough examination is beyond the scope of this work.

8.4 Implementation

Our vertical cache is implemented using several open source components along with our topic identification implementation. The cache is a simple web cache modulo our design changes implemented in approximately 2000 lines of Python code. The topic identification routines are approximately 500 lines of Python code. The content topic identification and prefetching algorithm are identical to the CIP implementation from the previous chapter.

8.5 Evaluation

In this section, we evaluate the effectiveness of our vertical caching layer using real-world access patterns of web browsing requests from our prior deployments.

In prefetching systems, the argument is that if there is bandwidth to spare then prefetching does not cost the user, and as a result there is an improvement in cache hit rate. The rationale behind a vertical cache is similar, but the notion of a "cache hit" needs to be revisited. In the literature, the typical method for evaluating such systems is to use traces and a bandwidth overhead quota to measure the prefetching benefits in cache hit rate or the resulting user perceived latency. Due to the different evaluation methodologies and traces between systems, some effort has been made to standardize this evaluation process [124]. However, evaluating a vertical cache based on cache hit is not particularly appropriate because vertical caching expands the functionality of the cache and relaxes the binary concept of a cache hit. Alternatively, measuring improvement in user perceived latency is trivial in offline mode, and does not capture whether pages returned by the cache are in fact relevant. We define a new metric "topic hit rate" to capture the idea of having a page in the cache that satisfies a request to some degree. A request results in a topic hit if there exists a topic identified by our system as defined previously.

8.5.1 Identifying Useful Topics

Using the log data from one school in Kenya, we show that we can extract useful topics. Overall, our log contains over 100000 requests and over 1400 search queries gathered over a period of four months. The usage of the cache is bursty due to multiple scheduling and classroom constraints, only 41 days out of 160 have any activity. Figure 8.1 shows examples of domain and query topics query topics and their cost on an arbitrary day.

Table 8.1: Example topics.

Topic	Cost
www.network-tutorial.com	201
msdn.microsoft.com	190
www.cisco.com	183
www.electronicsreviewsnow.com	114
imgs.xkcd.com	73
computer types	4139
system development methodology	19559
justin bieber	9
operating system development	14867
hoax virus	5

8.5.2 Topic Coverage and Cost Function

Figure 8.2 illustrates the percentage of topic hit rate of user requests as the number of domain topics chosen varies. The top domain topics in this figure are chosen are chosen by the highest coverage. Topics are defined in this case based on a threshold T requests belonging to a particular subdomain. We make three observations: First, relatively few topics are required to cover over 75% of the topics in our 4 month long log. Second, the resultant URL topic hit rate is insensitive to choice of the threshold T other than the tail of the graph is cut off. Third, the coverage is higher than the existing cache hit rate (66%) after approximately 40 domain topics. Topic coverage for query topics is defined in the same way. We constructed these topics from 1432 search queries, and find that after 10 query topics the topic hit rate is greater than the cache hit rate of 66%. As few as 50 query topics are sufficient to cover 75% of requests. The analog to byte hit rate of traditional caching is the percentage of cost of requests covered by the top topics, or the "topic cost hit rate". We find similar results with cost hit rate.

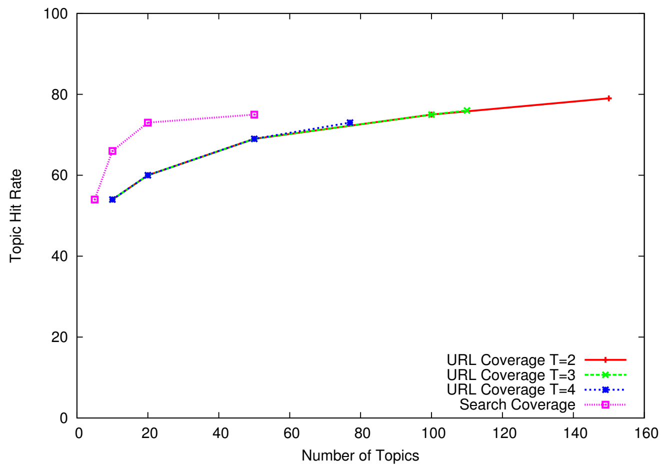


Figure 8.2: Topic hit rate of domain and query topics by request set.

8.5.3 Topic Recurrence and Stability

We next attempt to understand the utility of topics in another way: "how long does it take to detect a topic, and how useful is the topic over time?" Figure 8.3 shows a semi-log plot of the cumulative number of requests over time (Note: time here is in hours after the first request belonging to the topic) for a few of the top topics by topic coverage. The threshold $T = 3$ is shown for reference. Again, we make some observations. First, we observe that most of the domain topics are identified within the first 24 hours. Second, the identified topics recur regularly over time. This figure also shows that while requests are bursty over short timescales of hours, topic stability is fairly regular considering that the overall usage of the system is bursty (i.e. The long quiet periods are not necessarily due to lack of interest, but lack of overall use). We do not show query topics here since by definition query topics are constructed for queries within a certain time frame of the initial query.

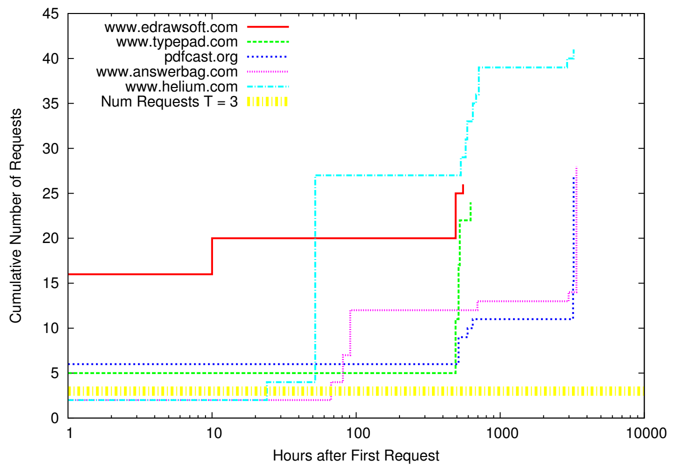


Figure 8.3: Semi-log plot of domain topic trends over time. Note that the x-axis is the number of hours after the first request for a topic and therefore not all topics extend to the end of the graph.

8.6 Related Work

Since we use the same techniques as in our CIP design, the work related to our vertical caching layer concerning feature extraction and focused crawling are detailed in the previous chapter.

Value-based caching by Rhea et al. [225] has a conceptually similar idea as vertical caching. In value-based caching, data is cached based on its value rather than its name(URL) so as to be more efficient due to the large amount of aliasing in web content. In vertical caching, instead of exact matching between chunks of data, we consider caching based on the information between in pages.

There are only a few other works on caching for emerging regions. Collaborative caching over delay tolerant networks [151] has been observed to improve cache hit rates dramatically due to similar interests across nearby villages. More recent work [152] extends their solution to support efficient Internet access for mobile users in Nicaragua. HashCache [90] is a highly efficient low-cost cache storage for emerging regions that is configurable to use no main memory for indexing.

8.7 Chapter Summary

In this chapter we designed a vertical caching layer for low bandwidth networks to maximize the utility of cached pages. Our system re-introduces the concept of topics that we assumed were inputs to CIP generation in the previous chapter. In this chapter we showed how to extract topics from web access patterns and contents of downloaded pages. We introduce a new cost metric per object based on time spent downloading to reflect both the actual cost incurred by downloading an object and also the time a user is willing to wait for the object. The cache contents are then organized based on these topics and costs for navigation, and caching mechanisms such as evictions operate at the per-topic granularity. Finally, we showed how topics and our cost metric may be synergistically applied to prefetching.

Chapter 9

A New Web Architecture for Emerging Regions

In previous chapters we examined the many different types of poor connectivity in emerging regions. For each of the connectivity scenarios we designed appropriate solutions and tested them both in the lab and on the ground with real users in their natural environments. In this chapter, we use our work to motivate and outline our design for a complete web architecture generalizable to any poor connectivity scenario. We design an end-to-end system for addressing all types of poor connectivity that consists of three main ideas. First, we incorporate *low-bandwidth transport optimizations* that significantly enhance the throughput and fairness of individual TCP flows and web session flow pools by using a combination of flow aggregation, loss-aware admission control and fine-grained packet scheduling. Second, we use our new *vertical caching* layer to organize our web cache based on user-centric *topics*. Third, our system streamlines user experience with an *asynchronous web browsing* interface that takes advantage of our transport and caching components.

9.1 Motivation

As we have shown in previous chapters, there are a multitude of problems that affect web access for each connectivity scenario. We summarize our results here to highlight the issues with the conventional web access model and motivate the need for re-architecting the web.

In Chapter 2, we identified serious deficiencies in web access and the web experience in two case studies in India. In Chapter 3, we illustrated that some of these problems are actually systemic to TCP. In the case of pathologically shared networks TCP approaches what we call the sub-packet regime and collapses.

In Chapter 4, we designed a middle-box solution to the sub-packet regime problem, but our techniques are incompatible with both TCP and HTTP standards. Furthermore, our system breaks with the conventional "always on" web interaction model. While packet prioritization is relatively transparent to the user, admission control is obvious to any user with a flow pool denied by a middle-box.

In Chapter 5, we showed that in slightly congested settings simple end-host web optimizations are an easy fix while requiring only browser optimizations. In Chapter 6, we studied networks that were constrained by high congestion and intermittency common in emerging regions. We found that web interactions under these conditions requires changes to application semantics as the network degrades past a certain threshold, and that these changes should not be completely transparent to the user.

In Chapter 7, we introduced techniques to prefetch and cache large repositories of web pages, or CIPs, for disconnected environments given a set of desired topics. We found that while caching and prefetching are commonly used to enhance web browsing performance in relatively good connection situations, extremely aggressive caching and prefetching are also beneficial to intermittent and offline environments with relatively focused topics of interest.

In Chapter 8, we extended the aggressive caching and prefetching to low bandwidth networks. The side effect of these optimizations is a tradeoff of availability with freshness. These aggressive techniques are also not possible without decoupling the availability of content with that of the Internet connection and violating web caching and prefetching standards.

Taken as a whole, our research shows that the existing web access model is fundamentally inappropriate at various networking layers and there is a pressing need for re-architecting the web for emerging regions.

9.2 System Overview

Our system design targets three broad classes of network connections in emerging regions: (a) low bandwidth network connections; (b) intermittent network connections; (c) mostly disconnected network connections. The second class of network connections, the network intermittency is relatively short-term triggered by network failures or high usage costs. The third class is when the connection is very high-latency over delay-tolerant network links (such as mechanical backhauls).

System Model: The basic system model consists of end-users behind a poor connection to the Internet. Users may spawn several web sessions in parallel, which may compete with each other. Figure 9.1 provides an overview of two different deployment models of our system. The top model consists of two overlay middle-boxes at either end of the constrained link (client-side and server-side proxies); all end-user web traffic is routed through these proxies. In the bottom deployment model, we have a single control entity such as a proxy close to the end-users before the resource constrained link. Alternatively, no middle-box support may be available and possible components would be placed at the client end. In that case, all modifications would be

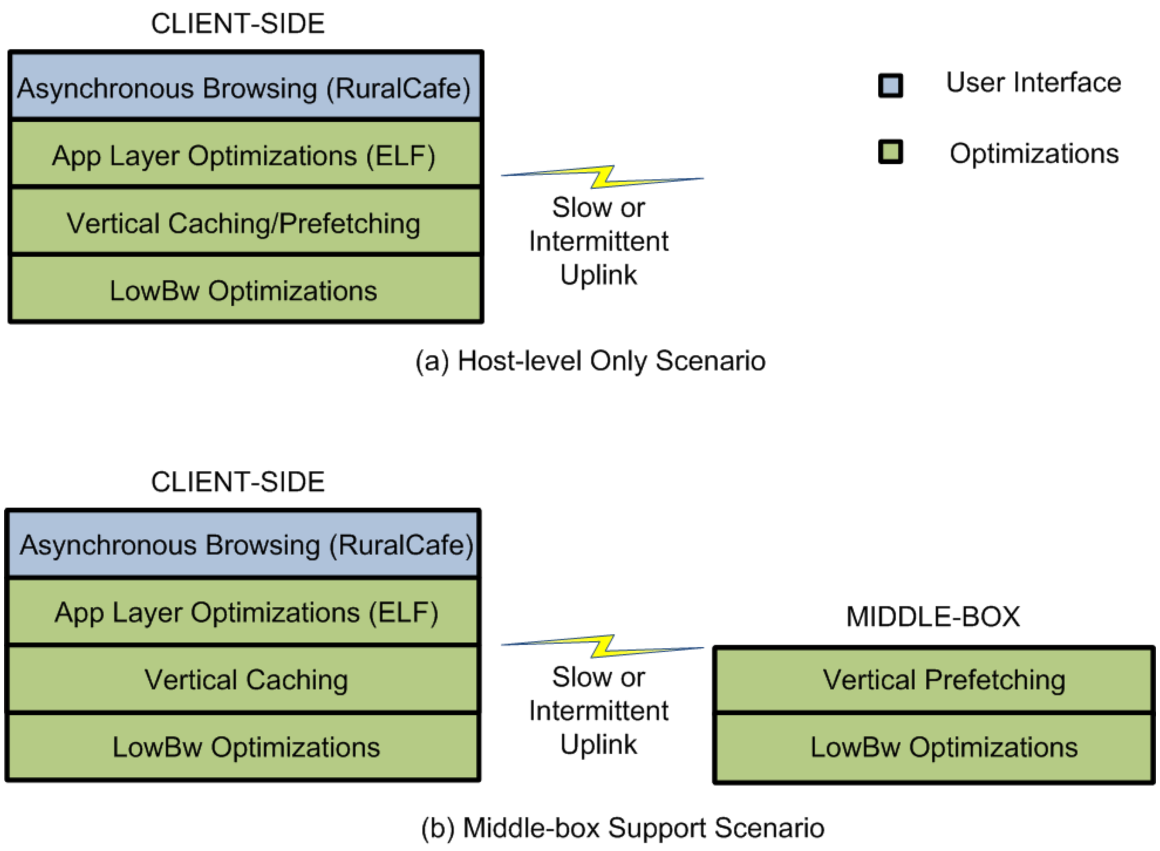

CLIENT-SIDE
Asynchronous Browsing (RuralCafe)
App Layer Optimizations (ELF)
Vertical Caching/Prefetching
LowBw Optimizations
Slow or Intermittent Uplink
User Interface
Optimizations
(a) Host-level Only Scenario
CLIENT-SIDE
Asynchronous Browsing (RuralCafe)
App Layer Optimizations (ELF)
Vertical Caching
LowBw Optimizations
Slow or Intermittent Uplink
MIDDLE-BOX
Vertical Prefetching
LowBw Optimizations
(b) Middle-box Support Scenario

Figure 9.1: Web architecture components.

made at the the client itself.

Our system consists of three basic components: (a) *Low bandwidth network optimizations;* (b) *Vertical caching layer;* (b) *Asynchronous web browsing.* Most of the functionality of the low-bandwidth optimization engine resides at the client side proxy. Our low bandwidth optimization layer primarily focuses on fixing the TCP breakdown problem. Other optimizations such as compression, pre-rendering, and filtering that have been proposed for low-bandwidth networks are easily incorporated into our architecture [28, 125]. The vertical cache also primarily resides on the client side, though the cache's contents can be updated by the remote proxy—either in bulk using a mechanical backhaul link, or incrementally using a low bandwidth network. Bulk updates for the vertical cache involve a vertical prefetching mechanism, primarily executed at the remote proxy under good connectivity conditions, and the results are updated using a mechanical backhaul link such as a USB stick. The mechanisms we employ are designed to be unobtrusive and compatible with existing intermittent systems.

Asynchronous web browsing: Every client is configured to connect through the client-side proxy where much of the intermittency-aware browsing functionality is implemented. We leverage the basic asynchronous query queuing model used in web search over delay tolerant network systems detailed in Chapter 5: users submit search requests, which the client-side proxy queues, and the proxy gives an immediate update on the expected response time for a query. The queue is persistent to the client, independent of the number of web browsing sessions the user activates; the client-side proxy can thus manage network resources and prevent self-induced contention. Recall, that in Section 2.3 we found that users in low bandwidth environments tend to open several browsing sessions hoping for better performance without realizing the self-induced contention it creates.

Our system behaves just like a normal transparent proxy when the network is available. Only as the connection deteriorates does our system gracefully transition to the asynchronous queueing model and becomes increasingly opaque. First, as the network becomes more congested, our admission controller begins rejecting new user requests. When a user request is rejected the proxy automatically switches the clients to “offline mode”. In offline mode, the topics are exposed and the user can interactively search and browse the contents in the cache. Similarly, users are also placed in offline mode when the network is actually disconnected. The user also has the option to manually switch to offline mode even if their if page load latencies are still too slow. In this manner, the web browsing experience is always interactive, the user is made aware of the underlying network intermittency when necessary, and the user is offered the flexibility to adapt their behavior as a function of the network conditions.

9.3 Implementation

The components of our architecture were implemented as they are described in previous chapters: End-host optimizations (Section 5.2), low bandwidth optimizations (Section 4.3), vertical caching(Sections 7.3 and 9.3), and asynchronous web browsing(Section 6.4). In addition, we leverage Solr [8], an open source enterprise search engine, to index and store the web page index meta-data for faceted browsing. For our faceted browsing and local search user interface we use Longwell [38], an open source faceted browser, during offline mode.

9.4 Related Work

There are very few works relating to architectures for emerging regions since it is a very nascent space. Only just recently Pai et. al. [203] proposed designing a new web architecture for emerging regions. Their work identifies a few of the same problem areas that we do, but our works are complementary and tackle different problem sub-spaces; our work can certainly leverage their results as an underlying network layer. They focus their efforts on optimizing performance of low-cost hardware [90], and WAN acceleration using peer-to-peer techniques [149]. In contrast, our work identifies problems with TCP, cache performance, and user experience. Work related to the components of our architecture are described in their respective chapters.

9.5 Chapter Summary

In our experiences deploying various systems in emerging regions we realized the many limitations of existing standards for accessing web content in these environments. TCP was not designed for sub-packet regimes, conventional caching and prefetching mechanisms were unable to preserve the "always on" access model of the web, and the inherent network *disconnectivity* required a new model for web access. Before coming up with a comprehensive solution we solved each of these problems independently. We designed our low bandwidth transport optimizations are to fix the TCP breakdown problem. Our vertical caching layer was designed to be extremely aggressive for flaky network connections. Finally, our intermittent web browsing system integrates these two components to provide an interactive web experience to the user. The resulting web architecture described in this chapter allows users to seamlessly transition between online and offline modes of web access. While further research will determine the most appropriate model for web access, our system is the first comprehensive solution we are aware of to the complex problem of enabling web access in emerging regions.

Chapter 10

SMSFind: A New Mobile Search Service

SMS-based mobile information services are increasingly common around the world, especially in emerging regions among users with low-end mobile devices. Web search is one of the most basic and useful of these services. SMS-based web search takes unstructured queries and returns web snippets via SMS. This application allows people with cheap mobile phones and no data plan to tap into the vast amount of information on the web. SMS-based web search is a challenging problem because the channel is both extremely narrow and expensive. Ideally, both the query and the response fit within an SMS.

This chapter presents the design and implementation of SMSFind, an SMS-based search system that enables users to obtain extremely concise (one SMS message) search responses for queries across arbitrary topics in one round of interaction. SMSFind is designed to complement existing SMS-based search services that are either limited in the topics they recognize or involve a human in the loop.

Given an *unstructured* search query, SMSFind, uses a conventional search engine as a back-end to elicit several search responses and uses a combination of information retrieval techniques to extract the most appropriate 140-byte snippet as the final SMS search response. We show that SMSFind returns appropriate responses for 57.3% of ChaCha search queries in our test set; this accuracy rate is high given that ChaCha employs a human to answer the same questions. We have also deployed a pilot version of SMSFind for use with a small focus group and a larger scale open beta in Kenya to explore our system and share our experiences.

10.1 Motivation

The exceptional growth of the mobile phone market has motivated the design of new forms of mobile information services. With the growth of Twitter [64], SMS GupShup [57] and other social messaging networks, the past few years have witnessed a growing prevalence of Short-Messaging Service (SMS) based applications and services. SMS-based services are also increasingly common in poor countries. Despite the increasing power of mobile devices with the advent of "smart phones", a significant fraction of mobile devices in emerging regions are still simple low-cost devices with limited processing and communication capabilities. Due to a combination of social and economic factors, voice and SMS will likely continue to remain the primary communication channels available for a non-trivial fraction of the population in emerging regions.

For any SMS-based web service, efficient *SMS-based search* is an essential building block. SMS-based search is a rapidly growing global market with over 12 million subscribers as of July 2008 [119]. An SMS message is constrained to 140 bytes, which drastically limits the amount of information in a search response. SMS-based search is also non-interactive due to the search response time; anecdotally [52], existing SMS-based search engines take on the order of tens of seconds [26, 77] to several minutes per response [13]. Even without the 140-byte SMS size constraint, tailoring traditional web search to mobile devices is a challenging problem due to the small form factor and low bandwidth. Unlike desktop search, users on mobile devices rarely have the luxury of iteratively refining search queries or sifting through pages of results for the information they want [237]. In this chapter, we address the problem of *SMS-based search*: *how does a mobile user efficiently search the web using one round of interaction where the search response is restricted to one SMS message?*

Though we do not know the internals of existing SMS search algorithms, we can observe from the user interface and documentation that existing automated services for SMS web search such as Google SMS [26] and Yahoo! oneSearch [77]) encourage users to enter queries for a number of pre-defined topics, or *verticals*. These pre-defined topics are either identified through the use of special keywords within the search query such as "define" or "movies" (e.g. Google SMS: "define boils") or have specialized parsers to determine which of the topics is intended (e.g. querying "AAPL" to Yahoo! oneSearch is a query for information about "stock quote"). ChaCha [13], a recent SMS-based search engine, hires humans to search the web and answer questions in an SMS response. TradeNet, now called eSoko [18], is a mobile marketplace platform in Ghana that would require an SMS based search service as well. Instant access to small bits of information is the motivation behind all of the existing SMS based search systems and of this work.

None of the existing automated SMS search services is a complete solution for search queries across arbitrary topics. Similar to traditional web search queries, SMS search queries suffer from the *long tail phenomenon*: there exists a long tail of search queries whose topics are not popular (e.g. "what are graduation gift ideas?" or "what chemicals are in a fire extinguisher"). We confirm that this indeed is the case in our sample of ChaCha questions where only 21% of the queries in our data set are verticals (as defined by Google SMS) and 79% are long tailed.

In this chapter, we describe the design and implementation of SMSFind, an SMS-based search engine specifically designed for the long tail of search queries that are spread across a wide range of topics. These topics represent the queries in a mobile search workload that are not answered by existing domain specific verticals. SMSFind is designed to integrate into an existing SMS search service to answer queries for unsupported long tail topics. Given a query, SMSFind uses a traditional search engine as a back-end to elicit several search results and extract the appropriate 140 bytes as the SMS search response. Section 10.5.2 further explains how vertical and long tail queries are defined in this work.

10.2 SMSFind Problem

SMSFind uses a combination of well-known information retrieval techniques to address the appropriate information extraction problem. SMSFind is designed for *unstructured queries* supporting a similar format as a standard search engine query. The key idea of SMSFind is that meaningful SMS queries typically contain a term or a collection of consecutive terms in a query that provides a *hint* as to what the user is looking for. The hint for a query can either be explicitly provided by the user or automatically derived from the query. SMSFind uses this hint to address the information extraction problem as follows: Given the top N search responses to a query from a search engine, SMSFind extracts *snippets* of text from within the neighborhood of the hint in each response page. SMSFind scores snippets and ranks them across a variety of metrics.

The architecture of our SMS search system (Figure 10.1) consists of a query server that handles the actual search query and results, and an SMS gateway that is responsible for communication between the phone clients and the query server. The client is a user with a mobile phone who sends an SMS message to the short code (a special telephone number, shorter than full telephone numbers used to address SMS and MMS messages) for our service, which arrives at the SMS gateway and is then dispatched to our server for processing. At our query server the query is then sent to a general search engine and result pages are downloaded. The query server extracts the results from the downloaded pages, and returns them to the SMS gateway that sends

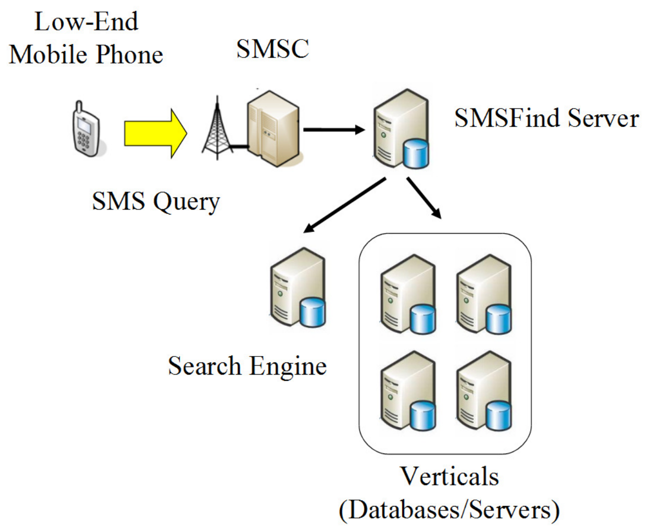


Figure 10.1: System architecture.

it back to the client that issued the request.

10.2.1 Known Verticals vs Long Tail

Certain SMS based queries are best answered by querying known verticals (e.g., flights, directions, weather). Google SMS search may be viewed as a wrapper around its search service for a fixed set of known categories such as phone numbers, weather, flight information, address etc. Our complete SMS search service consists of appropriate parsers and vertical redirectors for a few known categories (phone numbers, weather, addresses). For instance, *weather.com* and *yellow.com* are examples of such verticals for weather and yellow pages. For these categories with existing verticals, generating an SMS search response requires a simple transformation of the query into an appropriate database query (or filling a form) at the corresponding vertical web portal.

The focus of SMSFind is to handle long tail queries that do not fit into verticals. Handling queries for verticals is a relatively straightforward if tedious process and suffers from rapidly diminishing returns. Our complete SMS search service supports a basic set of vertical topics that is a subset of the Google SMS topics. As a part of this system, the SMSFind algorithm is placed behind a categorizer that first determines whether a given query belongs to an implemented vertical based on the existence of defined keywords or if it should be sent to SMSFind.

10.2.2 SMSFind Search Problem

SMSFind is designed for unstructured search queries across arbitrary topics. Given any query, SMSFind first uses an existing search engine as a back-end to obtain the top few search result pages. Using these pages the remaining problem is to parse the textual content to obtain the appropriate search response that can be condensed to one SMS message. This is essentially a problem of determining appropriate condensed snippets of text across the result pages that form candidate SMS search responses. We define a *snippet* as any continuous stream of text that fits within an SMS message.

SMSFind uses a hint for every query as an important clue to mine every result page for appropriate text snippets. A *hint* refers to either a term or a collection of consecutive terms within the query that is roughly indicative of what type of information the user is searching for. Given that the hint will appear in the response page, SMSFind uses the neighborhood of the text around the hint to mine for appropriate textual snippets. To explain our problem and algorithm we assume that the hint is given by the user explicitly, but later we discuss how the hint may be automatically extracted from natural language questions.

The SMSFind search problem can be characterized as follows:

Given an unstructured SMS search query in the form of <query, hint> and the textual content of the top N search response pages as returned by a search engine for “query”, extract a condensed set of text snippets from the response pages that fit within an SMS message (140 bytes) that provide an appropriate search response to the query. Where the hint is a term or collection of terms found within the query.

This problem definition explicitly assumes that the hint is specified for every query. Existing SMS-based search services like Google SMS have a similar explicit requirement where a keyword is specified as the last term in a query; the difference being that the existing systems only support a fixed set of keywords whereas SMSFind allows arbitrary hints.

10.3 SMSFind Search Algorithm

In this section, we describe our algorithm to extract snippets for any given unstructured query of the form <query, hint>.

10.3.1 Basic Idea

Search queries are inherently ambiguous and a common technique to disambiguate queries is to use additional contextual information, from which, the search is being conducted [133, 170]. Loosely, the term “context” is any additional information associated with a query that provides a useful hint in providing a targeted search response for a query [107, 175]. Similar to these works, SMSFind uses an explicit hint so the snippet extraction algorithm can identify the approximate location of the desired information in a search response page.

We motivate the use of a hint using a simple example of a long tail query. Consider the query “Barack Obama wife” where the term “wife” represents the hint. When we give this query to a search engine, most search result pages will contain “Michelle” or “Michelle Obama” or

"Michelle Robinson" or "Michelle Lavaughn Robinson" within the neighborhood of the word "wife" in the text of the page. For this query, to determine any of these as appropriate search responses, SMSFind will search the neighborhood of the word "wife" in every result page and look for commonly occurring *n-grams* (where n represents one to five consecutive words). For example, "Michelle Obama" is a n-gram that is a $2-$gram.

A simple algorithm to determine the correct answer to this example query is to output all popular n-grams within the neighborhood of the hint and rank them based on different metrics (frequency, distance etc.). However, outputting commonly occurring n-grams as search responses is only appropriate when the actual search response for a query is a $1-5$ word answer. For several common SMS-search queries, the actual appropriate search response is embedded in a sentence or a collection of few sentences. In such cases, we need to extract entire snippets of text as a search response as opposed to just n-grams.

SMSFind makes a clear distinction between *n-grams* and *snippets*. Though both represent continuous sequences of words in a document, a n-gram is extremely short in length ($1-5$ words), whereas a text snippet is a sequence of words that can fit in a single SMS message. In our SMSFind algorithm, n-grams are used as an intermediate unit for our statistical methods whereas snippets are used for the final ranking since they are appropriately sized for SMS.

We next describe the basic SMSFind algorithm. Consider a search query (Q, H) where Q is the search query containing the hint term(s) H. Let $P_1, \ldots P_N$ represent the textual content of the top N search response pages to Q. Given (Q, H) and $P_1 \ldots P_N$, the SMSFind snippet extraction algorithm consists of three steps:

Neighborhood Extraction: For each result page P_i, SMSFind searches for each appearance of the hint term H and extracts a textual neighborhood around the hint that represents a candidate snippet of length covering one SMS message in either side of the hint. For each snippet, we extract all unique n-grams with up to 5 words. The choice of the limit 5 is motivated by the fact that the Linguistic Data Consortium (LDC) [36] publishes the web frequency of every n-gram with up to 5 words.

N-gram Ranking: We rank the n-grams based on three metrics: distance to the hint, frequency of occurrence, and mean rank of the result page. We also use the relative rarity of a n-gram on the web to normalize the n-gram ranking.

Snippet Ranking: We define the rank of any snippet as a cumulative sum of the top-few ranked n-grams within the snippet. Among all snippets, we determine the top-ranked snippet(s) as the search response.

We now elaborate upon these three steps.

10.3.2 Neighborhood Extraction

Given a query (Q, H) and its result pages $P_1, \ldots P_N$, SMSFind first extracts snippets around the hint H in each of the pages. For each page P_i, we find every occurrence of the hint H in the page. Each snippet is up to 140 bytes long and the hint is as centered as much as the surrounding text allows. We found that delineating snippets by sentence boundaries lead to many corner cases due to noise that could skew the statistical results. The snippets are then merged if they overlap to avoid double counting into *snippet tiles* (Figure 10.2). These snippet tiles form the basis of all further measurements and calculations, and it is only within these snippet tiles that the final result is extracted.

From a practical standpoint of not needing to download several pages and having sufficient diversity to extract n-grams and snippets, we found that a value of $N = 10$ works well. We extract the text from these web pages by filtering out all scripts, hypertext tags, and non-ASCII symbols. The result is plain text that is similar to what would be rendered by a standard browser.

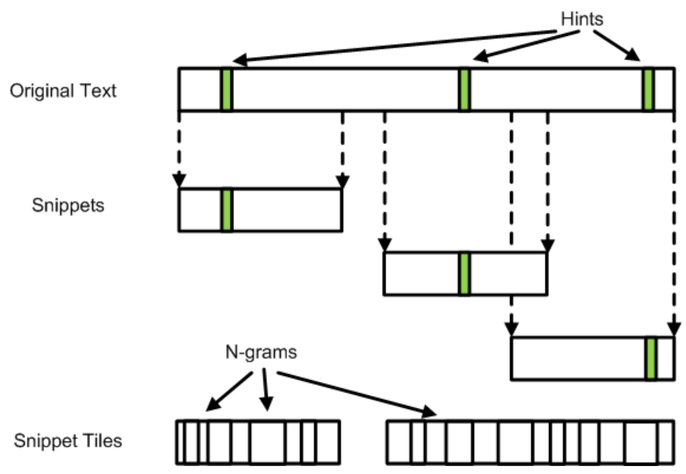


Figure 10.2: Snippet creation, aggregation into snippet tiles, and n-grams.

10.3.3 N-gram Ranking

N-gram ranking is a critical step in the SMSFind snippet extraction algorithm. Given any snippet extracted around the hint, the first step in our n-gram ranking algorithm is to gather all possible n-grams in the snippet. Table 10.1 illustrates briefly how the n-grams are generated. The goal of the n-gram ranking algorithm is finding the n-grams that are most likely to be related to the correct response.

The rationale of our n-gram ranking algorithm is that any n-gram that satisfies the following three properties is potentially related to the appropriate response for a query with a specified hint:

1. the n-gram appears very frequently around the hint.
2. the n-gram appears very close to the hint.
3. the n-gram is not a commonly used popular term or phrase.

As an example, the n-gram “Michelle Obama” is not a commonly used phrase and appears

Table 10.1: Slicing example for the text “the brown cow jumped over the moon”, hint = “over”.

N-gram	Frequency	Min. Distance
“the”	2	1
“the brown”	1	3
“the brown cow”	1	2
“brown cow jumped”	1	1
...	-	-

relatively frequently and in close proximity of the hint “wife” for the top search response pages to the query “Barack Obama wife”. Therefore, this n-gram is highly relevant for the search response for the query.

For each unique n-gram in any snippet, we compute three independent measures:

Frequency - The number of times the n-gram occurs across all snippets.

Mean rank - The sum across every occurrence of a n-gram of the PageRank of the page where it occurs, divided by the n-gram’s raw frequency. This is to incorporate the ranking system of the underlying search engine in our overall ranking function. (Some n-grams have a raw mean rank of less than 1 because the page containing the search results is assigned a rank of 0.)

Minimum distance - The minimum distance between a n-gram and the hint across any occurrences of both. Intuitively, this metric indicates the proximity of the hint defined by the user is to the search query. It is used as a part of our overall ranking function to allow the user hint to disambiguate two otherwise similarly ranked n-grams.

An example of the metrics we have at this point is shown in Table 10.2. In this example, the

Table 10.2: List of top 10 n-grams results for the query “the office dwight actor” and their associated raw metrics prior to normalization.

N-gram	Frequency	Minimum Distance	Mean Rank
wilson	16	1	1.5
rainn	16	1	1.25
rainn wilson	15	1	1.33
dwight schrute	9	2	0.78
schrute	9	2	0.77
actor rainn wilson	7	0	1.14
plays dwight	7	2	0.57
actor rainn	7	0	1.14
wilson who plays	5	2	0.8

query “the office dwight actor” should return the response “rainn wilson” as highlighted in the table. Note that this example is exactly in the range of queries that we are interested in, it is too rare for a custom extractor and common enough to be detectable by our system. From the list of n-grams we can observe that after slicing, most of the top results are highly relevant to the query according to our metrics.

Filtering n-grams: Before ranking n-grams, we filter the set of n-grams based on the three measures: frequency, mean rank and minimum distance. A n-gram should have a minimum frequency and should be within a certain minimum distance of the hint to be considered. For $N = 10$, we set a minimum frequency bound of 3 and a minimum distance threshold of 10; we choose these thresholds experimentally based on manual analysis of n-grams across sample queries. Similarly, we ignore all n-grams with a very low mean PageRank.

Ranking n-grams: Associating relative importance to any of the metrics or naively ranking based on a single metric is not appropriate. To combine the different metrics into a single metric we perform two simple steps. First, we normalize the raw frequency, mean rank, and minimum distance to a uniform distribution within the range between 0 and 1. We denote the three normalized scores of a n-gram s as $freq(s)$, $meanrank(s)$, $mindist(s)$. Second, the overall ranking score of a n-gram s is a linear combination of the three normalized ranks:

$$rank(s) = freq(s) + meanrank(s) + mindist(s) \tag{10.1}$$

We use the ranking score to rank all n-grams associated with a query. We need to consider one important detail in the normalization of the frequency score. If two n-grams s, t have the same frequency measure but if n-gram s has a much lower web frequency than n-gram t (s is rarer than t), then s needs to be higher ranked than t. We use the "Web 1T 5-gram Version 1" dataset from the LDC to obtain the frequency for any n-gram and compute its normalized frequency.

10.3.4 Snippet Ranking Algorithm

If the answer to a query is a very short response of a few words, then the best SMS based search response is to output all the top-ranked n-grams associated with a query. However, given a query, it is hard to determine whether the answer is embedded in a single n-gram or is actually a combination of multiple n-grams. In this scenario we output the best possible snippet under the hope that the answer is embedded in the snippet and the user can interpret it.

The n-gram ranking algorithm cannot be extended to ranking snippets since almost all of the snippets are unique, and their frequency measure will be 1. We extend the n-gram ranking algorithm to compute the rank of snippets based on the n-grams present within a snippet. In addition, different snippets may contain different number of n-grams that may introduce a bias in the ranking function. To avoid such a bias, we introduce a top-K n-grams based ranking function.

Snippet rank: Consider a snippet S with a set of n-grams $T = \{t_1, \dots t_m\}$ with corresponding n-gram ranks $rank(t_1), \dots rank(t_m)$. Let $t_{i_1}, \dots t_{i_K}$ represent the top K ranked n-grams in T. Then the rank of snippet S is:

$$rank(S) = \sum_{j=1}^{j=K} rank(t_{i_j}) \tag{10.2}$$

In other words, we define the rank of a snippet based on the cumulative rank of the top-K ranked n-grams within the snippet. In practice, we choose $K = 5$. In the *snippet ranking* phase, we determine the highest ranked snippet is the most relevant response to the original query. Recall that our snippets were designed to be under 140 bytes, but the snippet tiles may actually be longer depending on overlaps during the merging process. To find the best snippet, we first split each snippet tile into snippets using a 140 byte sliding window across each snippet tile that respects word boundaries. We then score each snippet based on the sum of the top K n-grams and return the top scoring snippet as the final result.

In the evaluation, we show that ranking n-grams first before ranking snippets is critical for better accuracy; in other words, directly scoring snippets from the web page results without using n-grams performs very poorly in practice.

10.3.5 Hint Extraction from the Query

We have thus far assumed that every query is associated with a hint, and for unstructured queries it is a natural tendency for the user to enter the hint either at the beginning or at the end. However, even if questions are entered in natural language format, extracting the hint automatically is not difficult. We describe a simple rule-based approach for deriving the hint for natural language question-style queries as a proof of concept. The approach we use is similar to surface pattern matching techniques [239].[1]

A cursory analysis of a sample of $100,000$ queries from ChaCha SMS-based search queries reveals that a large fraction of queries use common syntactic structures where the hint is easily identified. Nearly 95% of ChaCha search queries are English questions with 14 terms or less (75% of queries contain less than 10 terms). Based on the structure of the question, we have categorized a common class of questions and their corresponding transformation rules to determine the hint.

We found that that nearly 45% of the queries began with the word "what", of which, over 80% of the queries are in standard forms (e.g. "what is", "what was", "what are", "what do", "what does"). For each of these patterns, we can write a simple transformation rule to extract the hint from the corresponding sentence that is typically either immediately after the question or toward the end of the sentence. For example, for the query "what is a quote by ernest hemingway", the "what is X" pattern uses a simple matching rule to extract the hint term "quote" (if "a" is ignored as a stop word). The remaining terms minus stop words are then used as the query, and the final <query, hint> is: <"ernest hemingway", "quote">.

10.4 Implementation

The core SMSFind algorithm is implemented in only 600 lines of Python code and uses publicly available parsing libraries. We have not implemented any optimizations or caching, but SMSFind generally returns results within 5-10 seconds while running on a 1.8Ghz Duo Core Intel PC with

[1]More generally, determining what a natural language question is "about" has been studied, and there has been extensive work on question classification that is related to our problem [177].

2 GB of RAM and a 2 Mbps broadband connection. This response time is dominated by the time required to fetch query results from Google and download web pages referred to by the results. To deploy SMSFind as a search service we implemented a front-end to send and receive SMS messages. We setup a SMS short code with a local telco in Kenya, and route all SMS requests and responses to and from our server machine running the service across a Samba 75 GSM modem and Kannel an open source SMS Gateway [33]. As a proof of concept, and to improve the overall usability of our system we have also implemented interfaces to several basic verticals as a part of the service including: weather, definitions, local business results, and news. Each of these interfaces to verticals is under 150 lines of Python code.

10.5 Evaluation

Completely evaluating such a system is non-trivial; a large set of realistic queries must be asked, and answers must be judged either by judges or the users themselves. We derive a realistic set of queries by modifying real ChaCha queries. We then evaluate our system performance in absolute terms to explore the limitations of our system and potential improvements.

In our evaluation, the answers returned by SMSFind are judged correct *if the all of the correct answer terms appear anywhere in the returned snippet*. The correct answers were first decided by three judges who came up with correct answers by manual inspection of each question, referring to the ChaCha answer for reference. Sometimes even the ChaCha answer was considered incorrect by the judges and a different answer was determined online and used instead. There were also queries that had several acceptable answers, we consider all of those “correct” in our evaluation. This is the simple rating system we use throughout the evaluation. We do not currently rank our scores based on other important features such as readability or clarity. For a full-fledged question/answering system, incorporating mechanisms to improve readability [162] would likely be necessary.

10.5.1 Data

Our set of queries consists of a corpus of 100,000 real SMS search queries scraped from ChaCha's [13] public website on June 1, 2009. These queries are in Natural Language question format. In contrast to the logs analyzed previous studies we found in our data an average of 9.20 words per query and 47.09 characters per query, as compared to 3.05 and 18.48 respectively reported in [255]. The TREC [63] question/answering track datasets are based on search engine queries, and TREC-9 datasets were actual users' questions. However, it is unclear whether they were gathered from mobile devices. From previous studies it is evident that mobile search logs have significant differences across many characteristics depending on the search medium and device [159, 160]. Queries from ChaCha are mobile (either voice or SMS) and the answers are constructed by a human and returned via SMS. Therefore, we elected to use the ChaCha dataset due to its high realism for query types and their distribution across topics in a mobile setting. We only use all 100,000 queries for our aggregate analysis of query length and topic distributions. Since our evaluation consists of both rewriting, and manual judgement of queries and responses, both of which are labor intensive, we were unable to perform detailed evaluation with the full set of 100,000 queries. Instead, a randomly selected 1,000 query subset is used for our more detailed manual evaluations. Out of these 1,000 queries, we found through manual analysis that 793 are long tailed.

10.5.2 Query Topics and Identifying the Long Tail

In most query log studies, the queries are categorized according to topic to give a representation of the types of queries being submitted. Although this type of categorization is useful for that purpose, for our evaluation these categories do not help in deciding whether queries are part of the long tail of queries we wish to evaluate. Figure 10.3 illustrates the published questions

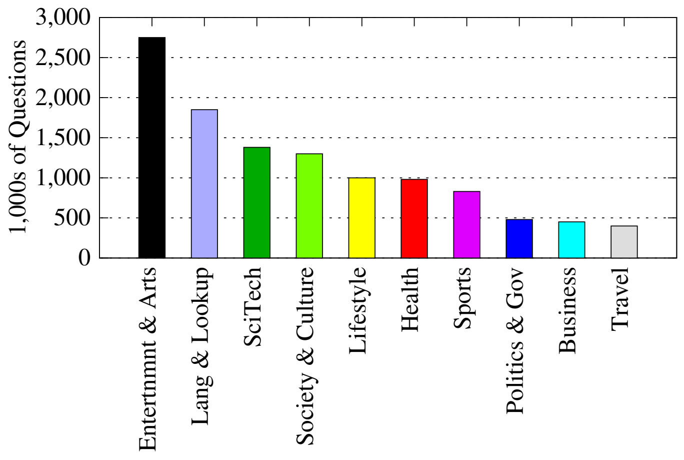


Figure 10.3: ChaCha number of questions per category as published on ChaCha's website [13].

per category on ChaCha's website [14]. This system of categorization by topic is similar to previous studies [159, 160, 255]. From these categories it is unclear whether topics such as "Entertainment" and "Travel" could map to verticals or are too broad and should be directed to our system.

Further dividing topics into sub-topics (using ChaCha's published breakdowns) reveals that some of these finer granularity of sub-topics are directly mappable to verticals: e.g. "Yellow Pages" and "Definitions" may be mapped easily to data sources at "yellowpages.com" or "dictionary.com" with little effort. Other sub-topics are still too broad for a simple vertical (e.g. the sub-topics of "Politics & Gov." such as "Law" or "World Governments").

Each question from ChaCha has an associated set of topics it belongs to as assigned by ChaCha. Table 10.3 lists the top 10 most highly represented sub-topics that identified in our sample (103 topics total), the percentage of questions belonging to these sub-topics, and whether there are corresponding verticals that are implemented in any existing SMS search systems. The sum of the percentages of the complete table do not add up to 100 because questions may

Table 10.3: ChaCha top sub-topics by percentage and existence of potential verticals.

Topic	% of total Questions	Existing Vertical?
Celebrities	10.6%	no
Movies (not show times)	9.0%	no
Yellow Pages	8.9%	yes
Definitions	8.1%	yes
Music	8.5%	no
Relationships	7.1%	no
Food & Drink	5.4%	no
Conditions & Illness	4.8%	no
Animals & Plants	4.6%	no
Games	4.5%	no

belong to more than one topic. Certain topics such as “Celebrities” have online databases such as Internet Movie Database (IMDB) that could easily be used as a source of structured information to answer queries, and implementing verticals on top of these would not be difficult. Using the sub-topics, we removed queries from our 1,000 query corpus that do not have existing verticals in any existing automated SMS search system to identify the long tail queries (793 long tail queries).

These remaining queries are exactly long tail questions SMSFind is designed to answer. Several examples are listed in Table 10.4. The question distribution across sub-topics and the diversity of questions in general suggests that mobile user information needs even on low-end devices are more diverse than previously demonstrated [160]. This may be due to the more expressive input modality of voice or the user’s perception of the system as being highly intelligent. We defer a more detailed comparison between these mobile search patterns and those in the literature to future studies.

10.5.3 Baseline Evaluation

We conduct a few baseline evaluations of our system using our sample of 1,000 queries from ChaCha containing both vertical and long tail queries. We modify these queries assuming the user understands how to use the system and is willing to enter keywords along with a hint term rather than a natural language question. For each ChaCha query we rewrite the query solely by removing and reordering words given that we have knowledge of how the system works and know what would likely be a good choice for the hint term. As an example, “what are the symptoms of chicken pox” is rewritten as “chicken pox symptoms” and the entire query is submitted to Google with “symptoms” used as the hint term.

Using these queries we find that our system (including redirection to existing verticals) results in 57.3% correct answers. In comparison, Google SMS returns correct results for only 9.5%

Table 10.4: Example questions from ChaCha (including spelling errors).

Question
“who do think is gonna win the nba game tonight cavs vs magic”
“what does tipo mean in spanish”
“what is the difference between hot and cute in a girl”
“what is the purpose of a widows peak”
“does the subaru wrx still come in hatchbacks”
“can you list all beastie boys albums chronigically”
“when they give you the death penalty what do they in sect you with”
“what and when is the highest scoring basketball ever in the nba”
“why does the rainbow represent gayness”
“is there eternal life”

Table 10.5: Summary of SMSFind results.

Input, Output	% Correct
Mixed, Snippets	57.3%
Long Tail, Snippets	47.8%
Long Tail, N-Grams	22.9%
Long Tail (w/hint), TF-IDF Snippets	20.2%
Long Tail (w/out hint), TF-IDF Snippets	5.2%
Long Tail Unmodified Queries, Snippets	16.1%
Long Tail Unmodified Queries, N-Grams	6.7%

of these queries. We note that the low performance of the Google SMS could be due to a variety of reasons that we discuss in Section 10.7, and the result is provided here only for reference.

What is more interesting is if we remove the queries that are redirected to existing verticals. To focus on the core SMSFind algorithm, we consider only the performance of the SMSFind algorithm on the 793 long tail query subset. All further evaluation in this section is conducted with this set of 793 long tail queries. Table 10.5 summarizes the results of our evaluation. We find that for the long tail queries SMSFind returns 47.8% correct results. Furthermore, if we consider only the highest n-grams returned rather than the entire snippet, the performance drops to 22.9%. These results broadly indicate that the raw performance of SMSFind has significant room for improvement, and returning snippet answers generally results in better performance.

We observed earlier that there are issues with readability, but what do the snippet results actually look like? A representative set of examples is shown in Table 10.6 for both correct and incorrect results. Compared to the ChaCha human written responses we observe that the readability of our snippets is poor and could be improved; however, finding the optimal snippet construction is a separate problem where existing techniques could be applied [162].

10.5.4 Is Distillation Using N-grams Beneficial?

Since we are returning snippets rather than n-grams, it is natural to ask whether n-grams are necessary or if ranking snippets alone would perform just as well. To confirm that the intermediate distillation stage using n-grams is beneficial we compare against a simple per-term TF-IDF approach. In the first portion of this experiment we gather SMS sized snippets directly using a 140 byte sliding window across all documents. In this second portion of the experiment, we do the same thing except the snippets (d_j) are scored based on the sum of the standard TF-IDF scores as the metric for each of the i terms (t_i) of the query.

$$(tf\text{-}idf)_{i,j} = tf_{i,j} \times idf_i \tag{10.3}$$

where

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}} \tag{10.4}$$

With $n_{i,j}$ is the number of occurrences of the considered term t_i in snippet d_j, and the denominator is the sum of number of occurrences in all terms in snippet d_j, and

$$idf_i = log\frac{|D|}{|\{d : t_i \in d\}|} \tag{10.5}$$

With $|D|$: total number of snippets and
$|\{d : t_i \in d\}|$: number of snippets where the term t_i appears

We find that with only TF-IDF, the correct result is returned in only 5.2% of queries. The results of the per-snippet correctness are shown in Table 10.5 for comparison. With the hint term used in conjunction with TF-IDF 20.2% of queries are correctly returned. Overall, both naive

Table 10.6: Example questions with snippet answers returned by our system.

Original Question	Example Snippet Answer	Judged Correct?
“how many packs of cigarettes are in a pack”	“to my contacts block user 10 packs and 200 cigarettes in a carton 3 years ago 0 rating good answer 0 rating bad answer report”	“yes”
“what does satine die from in moulin rouge”	“for a motion picture television or other visual media edit soundtrack main article moulin rouge music from baz luhrmann’s film songs”	“no”
“what rush album does natural science come from”	“seventh album permanent waves and the last of their by the way that really does kick *ss cygnus-x1.net a tribute to rush album review”	“yes”
“what is the least expensive state to live in”	“the wholesale price which is adjusted quarterly cigarette tax several states are continuing to raise excise taxes on cigarettes and other”	“no”
“what is the world record for the most hot dogs eaten in ten minutes”	“contest results 2008 nathan’s hot dog eating contest 10 minutes friday july 4 2008 no name hot dogs 1 tie joey jaws chestnut u.s”	“no”

TF-IDF and TF-IDF with a hint term return worse results than with the intermediate distillation using n-grams.

10.5.5 Returning Multiple Snippets

It is conceivable that for the queries that are not answered well, such as vague queries or explanations, slightly longer snippets or more results could improve our results. The application may allow multiple SMS messages to be requested in the form of a “more” link. To explore this possibility we experimented with returning multiple snippets. We compare two different snippet selection methods to improve result diversity. In both methods, we first order the snippets by rank. The first method is to return a snippet for each of the top ranked n-gram results. The second method returns only snippets for the top ranked n-gram result, but requires that the returned snippets are from different occurrences of the hint. We find that as more results are returned, the number of queries answered increases slightly (1 - 5%) for each additional response returned. Both methods show a similar rate of improvement, but the first method of maximizing n-gram diversity consistently performs better by approximately 10%.

10.5.6 How Important is the Hint Term?

One meta question we have considered briefly in our evaluation is: *can people actually come up with hints that are useful in the first place?* Requiring an additional hint term is “acceptable” since existing services by Google and Yahoo do exactly this, so our user interface is at least no worse than the norm. However, it is interesting to see how sensitive our system is to the existence of a well specified hint term.

We observe that if the hint term is not properly identified, and arbitrarily assigned as the last word in the query, the correctness of the snippets drops from 47.8% to 16.1% when compared to the modified queries. Table 10.5 shows the performance of our algorithm on our queries without any hint term identification or query modification for comparison. This suggests that the hint term is important to our system and a generally useful mechanism for focusing on the desired snippets of information.

10.6 Related Work

There has been relatively little research on SMS-based search. In this section, we take a slightly larger view of the problem space and contrast SMSFind with mobile search services, question/answering (Q/A) systems from the Text REtrieval Conference (TREC) community [61], and text summarization techniques.

10.6.1 Mobile Search Characteristics

Mobile search is a fundamentally different search paradigm than conventional desktop search. Yet, we continue to view mobile web search as either an extension of the desktop search model for high-end phones (such as PDA/iPhone) or a slightly restricted version via XHTML/WAP on low-end devices. Mobile search in both of these settings differs from traditional desktop search in several ways as shown in recent studies by Kamvar et al. [159, 160]. The first study [159] found that the mobile search click-through rate and the search page views per query were both significantly lower in comparison to desktop search. Meaning, most mobile search users tend to use the search service for short time-periods and are either satisfied with the search engine snippet responses or do not find what they were looking for. The study also found that the persistence of mobile users was very low indicating that the *vast majority of mobile searchers approach queries with a specific topic in mind and their search often does not lead to exploration.* The second study [160] showed that the diversity of search topics for low-end phone users was much less than that of desktop or iPhone-based search. This result suggests that the information needs are broad, but are not satisfied by the information services available on low-end phones. We find corroborating evidence in our analysis of ChaCha queries that mobile question diversity across topics is high given a more expressive input modality such as voice. As a whole, these studies indicate a pressing need for rethinking the current mobile search model for low-end mobile devices.

10.6.2 SMS-based Search Services

SMS-based search is very different from conventional mobile search via XHTML/WAP. An attractive aspect of SMS-based search is the lower barrier to entry of SMS (in comparison to other data services) due to the use of low-end phones and widespread availability of SMS. In poor countries, SMS is the most ubiquitous protocol for information exchange next to voice. In addition, there is speculation that "the economic downturn will most likely dampen growth for the more luxury-based mobile services, but SMS is expected to continue its growth as it is popular, cheap, reliable and private." [24].[2] Many of the top search engine players like Google [26], Yahoo! [77], and Microsoft [73] have entered the SMS search market and developed their own versions of SMS-based search services. All these services have been tailored for very specific topics (e.g. directory, weather, stock quotes) and specialized in nature.

These automated services are not the only ones available. ChaCha [13], Just Dial [32] (in India), and Google Baraza (in Africa) [25] are SMS- or voice-based question/answering systems using a human to respond to queries. The queries to ChaCha and JustDial are in natural language question form, interpreted by a human who looks for the solution online, and responds with an answer via an SMS. A recent private study by mSearchGroove [52] comparing the accuracy of responses of these services has shown that the automated systems suffered from low accuracy

[2]Most notably by Nielsen [119], but also by other private market research firms [24].

when satisfying an arbitrary set of queries: Google SMS 22.2%, Yahoo! oneSearch 27.8%, as compared to 88.9% for ChaCha. However, the study also observed that the median response time for the automated services was on the order of 10-14.5 seconds, whereas the median response time for ChaCha was 227.5 seconds. Involving a human in the loop drastically increases response time.

The problem we seek to answer is how do we build a system that achieves the best of both worlds: an SMS search system that is both a fast (automatic) and provides accurate query responses? One reason this problem is hard is that search queries are inherently ambiguous, yet returning a disambiguated result is especially vital to SMS search queries for various reasons [237].

10.6.3 Question/Answering Systems

The problem that SMSFind seeks to address is similar to, but distinct from, traditional question/answering systems developed by the Text REtreival Conference (TREC) [144] community. TREC has evolved into many separate tracks devoted to specific areas such as: simple ad-hoc tasks, question/answering tasks, million query tasks, etc. Nevertheless, our problem is different from each of the TREC tracks for at least one of three dimensions: (a) the nature of the input query; (b) the document collection set; (c) the nature of the search results in the query response. These three dimensions are derived by Carmel et. al. [103] in a model for assessing query difficulty.

First, from the input query standpoint, SMSFind is primarily suited for unstructured search style queries. SMSFind can be extended for simple natural language style quer-
ies using existing query transformation techniques as we discuss later. The distribution of the queries is also dependent on the mobile context (i.e. SMS or voice queries as opposed to Desktop search queries or iPhone queries). Second, SMSFind is a *snippet extraction* and *snippet ranking* algorithm that outputs condensed text snippets in existing pages as search responses. Third,

the most notable differences of our problem in comparison to many TREC tracks is that in SMSFind the collection of documents being searched over is a set of heterogeneous, “noisy”, and hyper-linked documents (web pages) indexed by Google. In contrast, the TREC ad-hoc track and TREC question/answering tracks have until recently used a collection of newswire documents that are very clean, and blog documents that have some noise [120]. Newswire and blog collections are also not hyper-linked and are several orders of magnitude smaller than the web. Prior work suggests that the size of the collection affects how well link analysis techniques may be useful to information retrieval [236]. More recently the TREC web, terabyte [118], and million query [81] tracks have used hundreds of millions of web pages as the document collection to allow for link analysis techniques to be applied toward the tasks, and there have been many systems that leverage the web for either the main or auxiliary corpus. The fact that the collection is at least two orders of magnitude greater than any TREC evaluation, significantly noisier in terms of information diversity, formatting, and being hyper-linked means that it is a different (and more challenging) problem, which leads us to adopt a different solution. The earliest and most closely related system to SMSFind in terms of architecture and approach is AskMSR [99]. AskMSR also leveraged the data redundancy of the web as opposed to sophisticated linguistic techniques to extract n-gram answers.

SMSFind shares techniques with prior systems [99, 117, 239], but is designed specifically for the unique requirements of SMS search in terms of its input and output. SMSFind expects mobile search queries along with a hint and returns snippets whereas existing systems may expect natural language questions and return n-grams or entire paragraphs and possibly the source document.

10.6.4 Automatic Text Summarization

Snippet extraction is also tangentially related to summarization, but only to the extraction task (finding sections of the text and producing them verbatim), and not the abstraction task (producing material in a new way). However, the main difference is the goal of these two problems. The goal of extraction is still to summarize the contents of one or more documents. The goal of snippet extraction in SMSFind is to find the correct answer to a query. For this reason, our problem is more similar to information retrieval. That said, many of the of the techniques are used in both areas of research including: Naive-Bayes methods [89], Rich features [178], and other statistical methods.

10.7 Discussion

The design of our system and the evaluation was motivated by the SMS search problem. Our goal was to understand how existing techniques would perform in this problem domain and not to supplant or reinvent the vast amounts of research in the Information Retrieval (IR) space.

10.7.1 Data Sets

We had initially experimented with Google Trends queries and other desktop search logs, but found that most queries were too ambiguous and did not match the type of queries found in mobile environments. We decided that the ChaCha dataset would provide more realism as to the actual queries people would make. To get a sense of the query difficulty and confirm that the ChaCha queries are actually "precision oriented" we performed a cursory evaluation based on work by Mothe and Tanguy [193] to evaluate the Syntactic Links Span and Average Polysemy Value of the queries in our corpus compared to those used in various TREC tracks. We confirm that for these two features our mobile questions are significantly different from those in the question/answering track. The closest match for these two features was the set of "search query" format queries from the Million Query Track.

10.7.2 Difficult Types of Queries

In terms of the performance of our system, we observed that the queries that are not answered properly regardless of format are often ambiguous, explanations, enumerations, problems that require analysis, or time-sensitive queries. This is not surprising as many statistical techniques have similar limitations. Our algorithm in particular requires proximity to the hint term and we also expect the answer to be only a few words long. Queries that are ambiguous such as: "how many calories are in baked cod" are ambiguous in the sense that the human intelligence was required to assume that Long John Silver's baked cod is nearly equivalent to "baked cod". *Explanations* are likely to require more space to answer properly. We observe that queries such as "how do i make an antenna for my tv? i have a wire hanger" are unlikely to be answered within 140 bytes by even a human intelligence. *Enumerations* such as "words that rhyme with long" are also difficult for SMSFind since the answer is unlikely to all appear consecutively near any hint term in a significant number of places on web pages. *Analysis* questions (e.g. "what is the least expensive state to live in") are problematic since the answer is not immediately available in the text of web pages. *Time sensitive* information is difficult for a combination of lack of web page sources and proximity to a useful hint term. Given the ubiquity of these difficult query types, it is somewhat surprising that our simple algorithm is able to answer over half of the dataset.

10.7.3 Comparison to Existing Systems

The inclusion of Google SMS performance on our data set is only to illustrate that the information needs of people using only voice and SMS to ask questions may not be captured by that particular system of verticals. A direct comparison between the two numbers is unreasonable since we have no information as to the distribution of queries Google SMS receives and how that compares to our test set. It is entirely possible that the distribution of Google SMS queries is different, and their verticals successfully satisfy a large percentage of queries received by their service.

10.7.4 Natural Language Processing and Mobile Craigslist

Combining statistical and linguistic techniques in the context of answering questions using the web has been thoroughly surveyed by Lin et. al. [179]. As part of future work, we plan to incorporate more sophisticated statistical and Natural Language Processing (NLP) techniques. Specifically within the NLP summarization literature, there are probabilistic models that learn how important various factors of a given shingle are for text summarization [89]. One fundamental distinction in SMSFind in comparison to NLP techniques is that the corpus varies as a function of the search query; given a query, the corpus for SMSFind is the search pages returned by a search engine. Given a highly variable corpus, directly applying existing NLP-based probabilistic models may not be appropriate since the model should be a function of the query.

To better understand this contrast, we briefly describe the problem of SMS-based Mobile Craigslist where the goal is to provide SMS-based search responses for Craigslist queries. This is a problem we are currently exploring where probabilistic models derived from NLP are directly applicable since there exists a well-defined corpus for any given search category (e.g. cars, hotels, apartments). Using this corpus, we can use standard NLP techniques to learn the various features for each category and derive a succinct summary of each Craigslist posting for a category based on the features. Having a well-defined corpus substantially helps in improving the accuracy of summarization.

10.8 Pilots

We deployed our system with a small focus group of 40 users consisting of both urban slum residents and college students in Nairobi, Kenya. Our pilot lasted for two weeks, during which, users with low-end mobile phones and no data plan found the service to be the most useful. Verticals were not initially implemented in our pilot, and SMSFind was tasked with answering all queries. We found in our initial feedback session that people requested local business contact information and other simple verticals that were not being answered properly by SMSFind. To address this, we implemented several verticals to increase query coverage.

After the inclusion of verticals with specific keywords, we found that our users took time adjusting to the syntax of requiring the hint/topic at a specific position of the query. We are currently exploring possible improvements to both the user interface and performance, and we expect to extend our scale of deployment over the next few months.

After our small scale pilot we released SMSFind in a large-scale open beta in Nairobi, Kenya while working with Nokia Research Center as a proof of concept. Further results and directions following the deployment of SMSFind are on hold pending discussions with Nokia.

10.9 Chapter Summary

In this chapter, we have presented SMSFind, an automated SMS-based search response system that is tailored to work across arbitrary topics. We find that a combination of simple Information Retrieval algorithms in conjunction with existing search engines can provide reasonably accurate search responses for SMS queries. Using queries across arbitrary topics from a real-world SMS question/answering service with human-in-the-loop responses, we show that SMSFind is able to answer 57.3% of the queries in our test set. We also show how SMSFind is incorporated into an overall SMS-based search system, and demonstrate its applicability in an open public beta in Nairobi, Kenya. Although more powerful IR and NLP techniques are bound to improve performance, this work represents a foray into an open and practical research domain. Furthermore, our techniques may be applied to other constrained IR tasks over unstructured documents beyond SMS-based web search.

Chapter 11

UjU: Redefining the Application Stack for Mobile Services in Emerging Regions

Mobile users in poor countries use low-end mobile devices and have restricted data connectivity, but have information needs that require reliable and fully featured applications. This chapter presents the design and implementation of UjU, a mobile platform that enables users to develop new SMS-based mobile applications on top of a common platform. Given that the SMS channel is constrained to 140 byte messages, UjU is designed to support database-centric applications that express and operate upon information in structured formats. In UjU, specifying a new application is equivalent to configuring an XML schema. Apart from exporting a standard set of operations, UjU allows the developer to specify new application-specific operations as XML forms. To make efficient use of the SMS channel, UjU supports a semantic compression engine that leverages the structured nature of the information transmitted. UjU includes a simple reliability layer to cope with message losses and uses a user-centric consistency model to handle data inconsistencies. We have configured and tested UjU for several SMS-based applications and describe our experiences in tailoring UjU to develop five real-world applications in the areas of mobile microfinance and mobile healthcare; four of them have been deployed in Ghana and Mexico.

11.1 Motivation

According to ITU [30], the mobile cellular penetration rate in emerging regions is 49 per 100 inhabitants with roughly 246 million mobile subscriptions in Africa. However, the mobile usage is still low due to high usage costs with an average monthly cost of 23% of monthly gross national income per capita [30]. This is a significant cost given that a large fraction of the population earns less than US$2 per day. In addition, recent studies have shown that the mobile usage does not increase proportionally with increased income [205]. The same study found that mobile usage will significantly increase if the costs are brought substantially down, which suggests that cost remains the main factor for low usage levels.

While usage currently remains very low, the low purchasing power also forces users to buy very cheap low-end mobile devices, which cost roughly $20 per device [205]. Low-end devices come with very limited set of features with voice and SMS remaining the primary communication channels available from these devices. It is also unclear whether smart phones will ever be able to break the $40 barrier and be available as a cheap communication device for the low-end markets. The current and projected price points of smart phones that support extended data services are still higher than the purchasing power of rural populations in emerging regions. Also, the real need for these sophisticated devices is far from established.

Apart from basic communications, there is also a growing interest among mobile operators to expand the range of mobile services in emerging regions. Data coverage has remained relatively low in emerging regions due to low user densities and purchasing power. However, SMS is universally available in all areas with cellular coverage and has been widely used in emerging regions. As a result, SMS remains the primary data communication channel available to end-users with cheap devices. Recent work by Oliver [197, 198] has shown that SMS may be used as a transport for mobile applications despite lost SMSs.

Mobile banking and healthcare are two popular areas where the drive for new applications has been immense. Many mobile operators in poor countries have their own platform for mobile banking mostly operating over SMS or USSD; the popular ones being M-Pesa [40], GCash [23] and mCheck [41]. At the recent Mobile Health summit [44], nearly 400 different mobile health applications for emerging regions were presented.

Non-SMS mobile application platforms are also being developed. There are several data collection systems being developed for mobile devices and specifically for use in emerging regions [22, 31, 48, 254]. A common requirement across most of these platforms is the need for relatively high-end mobile devices and data connectivity to operate, both of which are rare in rural areas. While it would be possible for field workers to carry smartphones, data channels other than SMS would not be available in rural areas. Large scale projects involving phones for end-users would be restricted to cheap mobiles due to cost and SMS or voice due to network availability.

Operational sustainability is another crucial consideration for any SMS-based application since sending and receiving SMS messages are expensive. Projects based on phones are at the mercy of mobile operators to keep costs low enough to make projects sustainable. This is often not the case. In Malawi, health workers use FrontlineSMS to update patient records via SMS messages. The estimated HIV prevalence is 11.9% [65] of the 14 million person population [71], thereby accounting for approximately 1.5 million HIV-infected patients. The population covered by the mobile cellular network is 93% and one SMS message costs US$0.09. Assuming that the health workers send a single SMS message per month as an update per patient, the net communication cost of the system across all patients is US$135,000 per month. In the current program, a CHW visits five patients per day, requiring altogether 10,000 CHWs. The current salary of a CHW is around US$60. So, the total cost is US$(10000 * 60 + 135000)*12 = US$2.4M per year. While these figures are simply estimates, it is clear that the communication cost is non-trivial and operational sustainability is highly dependent on the bulk SMS rates dictated by the mobile operator.

Apart from operational sustainability considerations is the required expertise for application development and maintenance. Practitioners in emerging regions are interested in creating applications for their projects, but many practitioners are not computer programmers. Hiring competent engineers to develop custom applications is a relatively large monetary and time investment. To address these problems with mobile application development for emerging regions we designed UjU as a framework for quickly and easily creating new applications on cheap mobile devices without requiring programming expertise.

11.2 System Design

An UjU application is not a J2ME application. An UjU application is defined as a constrained set of operations that act on a database within the UjU framework. Therefore, UjU may be used to implement form-based applications using database-like primitives and not applications such as games or education tools.

We designed UjU around two goals:

1. Make creating SMS-based applications simple for people without without programming experience.

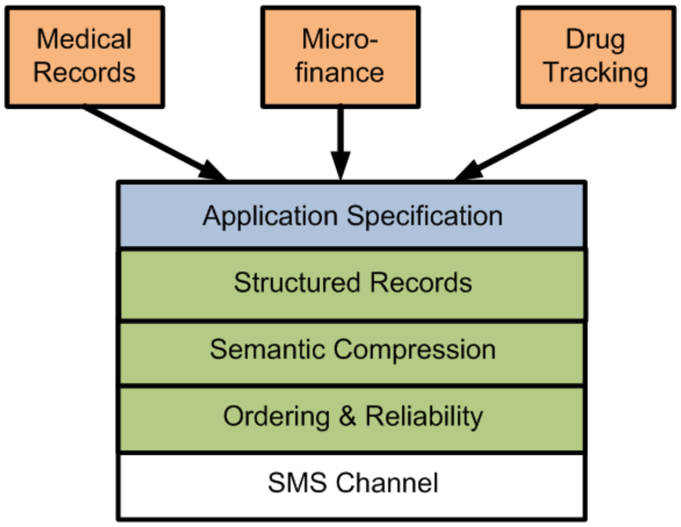


Figure 11.1: The UjU design stack.

2. Given the cost of an SMS message transmission and the 140 byte constraint of an SMS message, make SMS-based applications operationally sustainable.

As with typical mobile application designs, UjU assumes a centralized server communicates with users who use applications on their mobile devices to fetch, update, and search records on the central server. Figure 11.1 illustrates the UjU design stack. We designed UjU for applications that operate over structured data, which allows us to leverage structure for more efficient SMS channel usage. We also implement lightweight mechanisms to handle reliability challenges of SMS. Applications such as medical record systems, microfinance, and drug tracking may be implemented through the application specification to take advantage of the UjU stack.

11.2.1 System Components

To implement this design stack, UjU consists of four main components.

1. UjU *Application Specification:* In UjU, specifying a new application is equivalent to configuring an XML file; UjU allows users to create their forms using a simple web interface. Figure 11.2 presents the current version of the UjU web interface. UjU supports a standard set of data manipulation and search operations but also allows the user to create new operations using simple XML forms.

2. *Aggregation of Operations and Structured Records:* To enhance the channel utilization and minimize operational costs, UjU caches updates and aggregates multiple operations or records into a single session. Since intermittent updates may introduce data inconsistencies, UjU supports a flexible consistency model that allows administrators to handle inconsistencies. The default is an append-only model where every update is appended to a list and every fetch operation retrieves the last few updates from the append list.

3. *Semantic Compression:* To make efficient use of the underlying SMS channel, UjU supports a semantic compression layer that semantically encodes a structured stream and achieves a high compression ratio in comparison to standard compression techniques.

4. *Lightweight Ordering Reliability:* Given that the underlying SMS channel is unreliable, UjU supports a lightweight reliability layer that supports session-level reliability and uses delayed acknowledgments that are piggybacked in normal data messages.

11.2.2 Application Specification Made Easy

UjU is designed for supporting field level manipulations for structured records. UjU allows users to specify an application as an XML configuration file that captures the structure of records within an application and the parameters associated with each field. UjU supports two different types of XML configuration files: *Data Description* and *Form Description*.

Data Description: An *Data Description* XML file is the primary application specification file that defines the structured record schema and their associated data types. Each field, in the

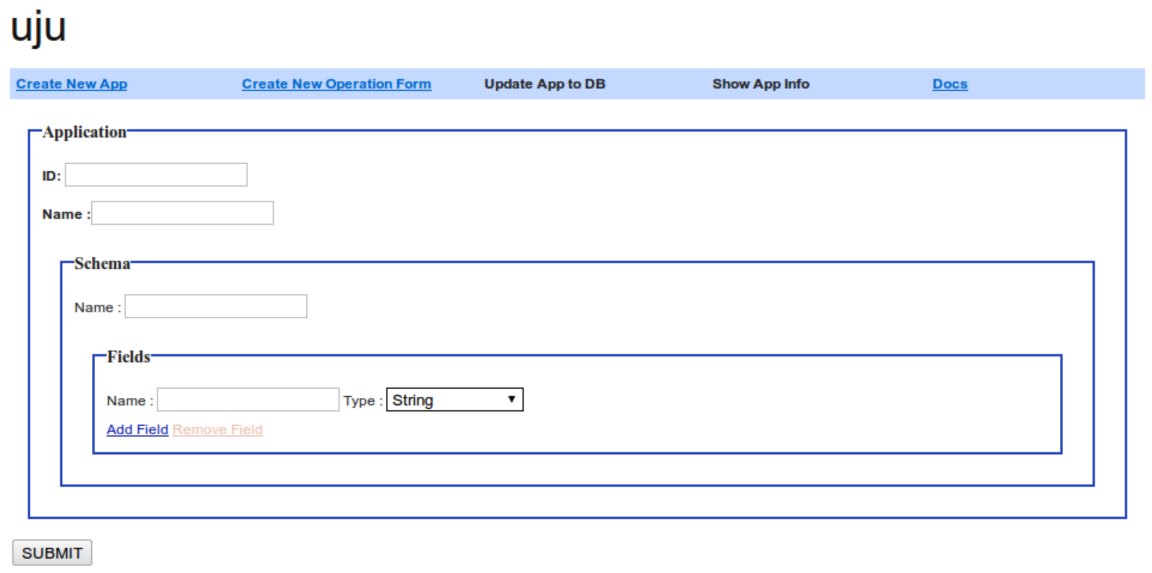


Figure 11.2: The current UjU web interface for creating an Data Description XML file.

schema, has three attributes: `Field ID`, `Name`, and `Range`. `Field ID` is a unique identify number, which will be referenced in the Form Description. `Name` of field will appear in the database and the screen of the client's mobile device. `Range` is a special field that is required for semantic compression. Figure 11.3 is an example of the *Data Description*. In this particular example, the application id is "0" and its name is "CAME". To allow several applications to be continuously used on top of the same platform, every application is associated with an application ID. This application has only one database schema, which has 9 fields. To further simplify specification, one can derive the XML specification from a simple web form.

Form Description: The *Form Description* defines a specific operation and the fields that relate to the operation. The operation may be visualized as a simple SQL statement as illustrated in Figure 11.6(a). Given any SQL operation, we can determine the corresponding form for that SQL statement. Figure 11.4 is an example *Form Description* for the SQL statement specified in Figure 11.5. In this *Form Description*, the id is **1**. This form belongs to a specified application

```
<?xml version="1.0" encoding="UTF-8" ?>
<application id="0" type="0" name="CAME">
 <schemas total="1">
  <schema id="0" name="client">
   <fields total="9" primary_field_id="0">
    <field id="0" name="Client ID" range="24"/>
    <field id="1" name="Group ID" range="16"/>
    <field id="2" name="Week’s Number" range="8"/>
    <field id="3" name="Saving" range="18"/>
    <field id="4" name="CAME Capital" range="18"/>
    <field id="5" name="Interciclo Capital" range="18"/>
    <field id="6" name="Default" range="18"/>
    <field id="7" name="Late Payment Times" range="8"/>
    <field id="8" name="Miss Meeting Times" range="8"/>
   </fields>
  </schema>
 </schemas>
</application>
```

Figure 11.3: Data Description XML file example.

with an identity number **0**. Every operator is associated with an identity and this operation has an id **3**.

UjU supports the following basic operations for any application: `Create`, `Update`, `Destroy`, `Fetch`, `Search`. The create and destroy operations allow users to create and delete records. The Fetch operation allows the user to fetch records that match a specific criterion. The simple fetch operation allows the users to pick a specific field and specify a constraint based on the field. The user interface also allows users to select multiple conditions and combine them using an AND or an OR clause. A user that needs a sophisticated fetch constraint needs to create a new XML *Form Description* for the operation.

The Search operation is a field-agnostic fetch operation in UjU. The search model in UjU is simple. Users can search for records based on a set of keywords where the search operation treats all fields as a text stream and outputs all records where the specified keywords appear in some field in the record.

Updates and Consistency Model: The update operation allows the user to update one or more fields in a given record where we explicitly assume that a record is identified by a specific identifier known to the user or has a secondary key based on name or alternative parameters as a unique key. Given that users may access the database in an intermittent fashion, UjU uses a default append-only consistency model where all updates to a given record are appended to an append list. UjU explicitly exposes the inconsistency at the server to the administrator who is responsible for reconciling inconsistencies. Fields within UjU can be marked as either *one-time modify* or *append-only* or *last-modify*. The one-time modify fields are set during record creation (primary key values) and the append-only fields are not modified but maintained as an append-list. Users can also set the application or individual fields to last-modify. This retains the last updated value independent of the order of updates. Append-lists are associated with a list of bounded length.

```
<?xml version="1.0" encoding="UTF-8" ?>
<form id="1" application="0" operation="3" name="Client Info">
 <table>
  <schemas total="1">
   <schema id="0">
    <fields total="3" logical="AND" >
     <field id="0" operator="Equal to"/>
     <field id="1" operator="Equal to"/>
     <field id="2" operator="Less than"/>
    </fields>
    <fetch total="6">
     <field id="0"/>
     <field id="1"/>
     <field id="2"/>
     <field id="3"/>
     <field id="4"/>
     <field id="5"/>
    </fetch>
   </schema>
  </schemas>
 </table>
</form>
```

Figure 11.4: Form Description XML file example.

```
SELECT `Client ID` `Group ID` `Week's Number`
`Saving` `CAME Capital` `Interciclo Capital`
FROM `CAME`.`client`
WHERE `Client ID` = 485080
AND `Group ID` = 17291
AND `Week's Number` < 5
```

Figure 11.5: SQL statement corresponding to the example forms.

Figure 11.6: We present a unified example for the how to create a simple application with the components of UjU: Data Description, Form Description, and the generated SQL.

11.2.3 Semantic Compression and Aggregation

The goal of semantic compression is to minimize the number of messages that UjU uses for the various operations communicating with the server. Unlike traditional compression algorithms, the basic idea of semantic compression is to derive a compression codebook to represent each field (or a group of fields) in its lowest entropy using the smallest number of bits.

Consider a specific field and the universe of values that the field can take. A semantic compression codebook will generate a specific code for every possible value occupied by the field. Given the relative frequency of occurrence of every value in the universe, a Huffman code is generated for the field that will represent the field using a minumum number of bits. For most applications, this frequency information is either not available or may change with time. An alternative simpler coding scheme is to used a fixed-length representation per field. If a field occupies n states, then the field can be represented using $\log n$ bits. If the universe of values for a field is unknown, we simply associate the field with a fixed length (with no compression).

While this is a relatively simple coded representation, we have also developed a joint-coding framework where the universe of fields is known or partially known. The basic idea is to measure the joint entropy of different fields and join multiple fields that are strongly correlated with each other. Joint entropy is recursively applied until individual fields are not correlated with each other; i.e., joint entropy is comparable to the sum of the individual entropies. Joint entropy coding is not enabled by default in UjU. Other compression algorithms such as arithmetic or delta coding across messages are also possible alternatives. We did not experiment with these coding schemes in our current implementation.

UjU simplifies the ability for the application to specify the universe associated with each field. UjU supports the following five basic data types: integer, string, single choice, multiple choice, set. Integers are optionally associated with a range value that automatically specifies the fixed-length code. Similarly, string is a fixed-length code. Single choice is specified by $\log n$ bits for the n choices and multiple choice is represented using n bits. The set function allows users to specify a universe of values that is used to derive the compression code. The universe of values is pre-specified in the application configuration file that the client and server use to derive the compression code.

To achieve better compression, UjU supports aggregation at both the operation level and the record level. The client maintains a queue of recent operations. When the queue has sufficient operations to fill a single SMS message, the client sends a batch of operations to the server. Similarly, multiple records are be compressed into a single message in the response from the server.

11.2.4 Lightweight Reliability

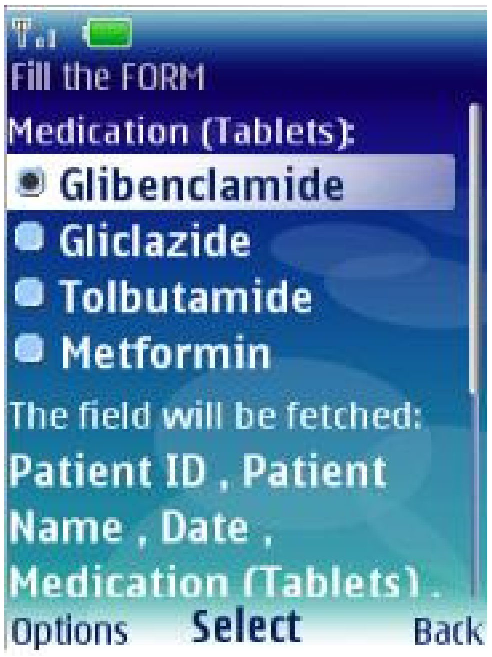
SMS message delivery is not 100 percent reliable and messages that do arrive are not guaranteed to be in order [197]. Recent work by Oliver [198] has proposed the use of SMS as a transport channel. Our lightweight reliability is a simplified version of the transport presented by Oliver.

We found that a reliability model similar to TCP/IP would be excessively complex and expensive over SMS. Protocols requiring multiple rounds of acknowledgements are impractical since each message incurs a cost. To minimize communication cost we propose a thin reliability layer.

UjU uses an intermittent communication model between the mobile device and the server. In this model messages are divided into *sessions*. A session is a two-way communication between the client and server. Most operations are client driven in UjU. During a session, the client sends a string of SMS messages to the server followed by a string of SMS messages in the reverse direction. Every session is associated with a 8-bit identity and a session supports up to 16 messages in each direction. Each message in either direction contains a 4-bit sequence number. The 16 message limit is an artificial bound we set to simplify our design.

UjU uses three communication phases in a session. In the first phase, the client initials a session with a new session identifier and sends an aggregate stream of SMS messages. In the second phase, the server responds with the responses along with 16-bit bulk acknowledgment along with the responses to the client operations. If the response does not fit within 16 messages, the server needs to initiate a new session. All messages are semantically compressed to reduce the number of messages. If any message is lost or received out of order, the client or the server waits for a short period of time, before sending a NACK message with the unreceived messages. If all of the initial messages is lost, the client waits for a short period of time and resends all of the initial messages. In the final phase after a round of interaction, the client can either immediately acknowledge receipt of the records to the server or can use lazy acknowledgements to provide an acknowledgement at the beginning of the next session. The lazy acknowledgement saves one additional SMS message at the client.

In general, our reliability layer is a batching based solution that does not guarantee perfect reliability but operates under the assumption that most SMS messages are correctly delivered. Under this protocol, a session may not complete only if: (a) all messages from client to server


Fill the FORM

Medication (Tablets):

● Glibenclamide

○ Gliclazide

○ Tolbutamide

○ Metformin

The field will be fetched:

Patient ID , Patient Name , Date , Medication (Tablets) .

Figure 11.7: An example application written on J2ME for UjU.

are lost; (b) all messages from server to client are lost.

11.3 Implementation

In this section, we describe some of UjU's implementation details. We elaborate on the nuances of our terminology as well as explore the client- and server-side details of UjU.

11.3.1 UjU Client

The primary role of UjU mobile phone client-side software is interacting with a remote database through SMS messages. Our client-side software is implemented as a J2ME application and has been tested across different Nokia phone models (Figure 11.7). UjU requires the server to specify a *User Data Header (UDH)* to enable the UjU client to receive the SMS message; if not specified, the mobile device will place the SMS message into the normal message inbox instead of passing the SMS message to the UjU client application. Considering the limited storage space on the mobile device, UjU stores only the critical information from the configuration file and does not store the entire XML file; this reduces the size of the configuration file and allows UjU to support many applications simultaneously. UjU uses the Record Management System (RMS) functionality in J2ME to maintain the configuration files and the individual local databases as row data. This allows UjU to store data in flash memory.

11.3.2 UjU Server

A deployed UjU server has the primary function of receiving SMS messages and, if necessary, sending response SMS messages.

UjU leverages Kannel, an open-source *Short Message Service Center Gateway*, or SMSC Gateway. It presents an HTTP and HTTPS request interface for applications to receive and send SMS messages. Depending on the type of SMSC, Kannel can use either AT commands to communicate with the GSM modem or a set of APIs/SDKs provided by the wireless carrier. Our current implementation uses a Samba 75 GSM Modem and Kannel uses AT commands to communicate with the modem.

The basic access model of UjU involves a client application running on the user's mobile phone. The client application accesses the remote database by sending and receiving SMS messages to and from the server's SMSC gateway.

The SMSC gateway distributes messages to the UjU server, which first decodes the messages. Decoding is necessary since most of the bits in the received SMS message are semantically compressed. Parse-able header information in the SMS message indicates to the server which

Data Description to use to semantically decompress the bits of the SMS. This Data Description information is fetched from the local "Data Description" database table and used to decompress the SMS message data. The final SQL query is made to the application's own database table based on the contents of the SMS message. If necessary, a return SMS message is prepared based on the original query and sent.

11.3.3 Application Creation

Creating an application consists of two steps: (1) creating the **Data Description** and (2) creating a **Form Description** for each desired operation.

To clarify the discussion on supported operations, we present the following definitions. A *transactional operation* refers to operations that affect the stored state of an application; e.g., updating a record on the server is a transaction operation. Create, Update, Fetch, Destroy, and Count are already implemented transactional operations. A *logical operation* is a function whose inputs are data and whose output is a single value; e.g., the sum of fetched numeric values.

Currently, our system comes with built-in basic transactional operations for creating new Form Descriptions. For future work, our system will automatically allow users to define their own logical operations with the web interface.

An *application* consists of a series of *input fields*. When an application writer specifies an application using our web interface, the web service automatically generates a downloadable XML file of the Data Description. A user may fully specify their Data Description using our web interface.

11.4 Evaluation

In this section, we present the evaluation of UjU across a set of sample SMS-based applications to illustrate the ease of application specification and the effectiveness of semantic compression. Specifically, we show that: 1) creating a simple application only takes 15 minutes and one can compress an application configuration file to be transmitted over-the-air in few SMS messages; and 2) UjU supports high levels of semantic compression and can aggregate several operations/records into a single SMS message.

11.4.1 Application Specification

Application Creation Time

	Number of Fields								
	Integer	Date	Generic String	Name	Multiple Choice	Single Choice	Boolean	Total	Create Time (minutes)
Patient Intake Form	19	2	12	6	0[0]	3[103]	0	40	55
HIV/AIDS Diagnosis Form	13	9	1	1	0[0]	5[15]	5	25	35
Used Car	5	1	1	1	1[12]	0[0]	1	9	18
Top 10 Digital Camera	2	1	1	0	0[0]	2[59]	0	6	30
Case Report	6	2	2	0	0[0]	1[2]	0	11	15
BMI	4	2	2	1	0[0]	1[2]	0	10	15

Table 11.1: Time for members of our lab to create an application on top of UjU.

We first evaluate the required time for creating an application using UjU. We present evaluation based on a collection of form-based applications used for a variety of purposes, such as patient records, Craigslist, and epidemiology. We list the time for creating applications in table 11.1. It only takes 15 minutes to create a very simple application for BMI records. This application only contains 10 fields, Patient ID, Sex, BMI value, etc. In addition, the configuration for the basic operations can also be automatically derived.

The creation time for an application is fully dependent on the number of fields and number of choices in the single or multi choice field. Taking Patient Intake Form as an example, there are only 20 fields in the application. However, there are 103 choices in this application. Thus, this application takes more than 55 minutes to create.

Size of Configuration File

	Data Description		Form Description	
	Client	Server	Client	Server
Patient Intake Form	1966	9075	376	2805
HIV/AIDS Diagnosis Form	1041	3990	338	2369
Used Car	359	1732	255	1963
Top 10 Digital Camera	924	4457	277	1886
Case Report	273	1425	203	1929
BMI	276	1372	200	1929

Table 11.2: Size of Configuration File (in bytes).

Mobile devices have limited storage space so the size of configuration files should be as small as possible. Given that low-end mobile devices do not export a standard database interface, UjU uses *RecordStore Management System*, or RMS, on the client side. RMS is part of the J2ME environment and allows developers to record data in flash memory. A configuration file is represented by a single XML file. In the XML file, developer defines an *Element* and appends several attributes belonging to that element. An example of elements in the configuration file of an application might be:

```
<type id="1" name="integer" range="17"/>
<field id="0" name="Patient"
       type="1"/>
```

While this representation potentially consumes a lot of space, we can use the same semantic compression technique to represent an application configuration file as well as the operator-level configuration files. Table 11.2 shows the total size of all configuration files for different applications in bytes. The memory footprint of the configuration files are extremely small and most applications and operations can be transmitted to the client (Client configuration file) using a few SMS messages. While simple applications can be transmitted to the client in 2-3 SMS messages, complicated applications with several entries may consume up to 15 SMS messages. Individual operations can be transmitted to the client in less than 2-3 SMS messages. This allows for a powerful paradigm where SMS applications can be installed over the air using only a few messages.

11.4.2 Semantic Compression

To evaluate semantic compression, we consider the same six applications and compute the number of bits required for representing a single operation and a response. Creating/returning an entire record is the most space-consuming task for both the client and server. Table 11.3 shows the size of a record (in bits) after semantic compression. Across all these forms generated from different real-world settings, UjU can condense each form roughly 200-400 bits per record. In the worst case, for certain large forms, UjU may consume close to 600-700 bits to represent an entire record. We note that the same forms represented in text format can consume a few KB even after Gzip.

Such a compact representation allows UjU to aggregate multiple records in a single SMS message. In the best case, UjU is able to transmit 8 records into one single SMS message. This level for aggregation represents a lower bound since these are create operations where the entire record is transmitted over the wire.

Most common operations fetch or update very few fields within the existing records. Under such conditions, UjU can support much higher compression levels. In the case where users are interested in fetching only specific fields, UjU allows users to easily specify these fields using

	bits/record	record/message
Patient Intake Form	674	1
HIV/AIDS Diagnosis Form	188	5
Used Car	207	4
Top 10 Digital Camera	116	8
Case Report	231	4
BMI	262	3

Table 11.3: Required bits per record.

	bits/record	record/message
Fetch Emergency Content	206	5
Fetch Patient Content	159	6

Table 11.4: Fetching fields.

a simple selection interface on their mobile device. We evaluate this using the Patient Intake Form where we consider two different **fetch** operations will only fetch the patient important medical information or emergency contact information (along with basic patient information). Table 11.4 shows the result that UjU can accommodate 5-6 records in a SMS message including the protocol header bits. Compared to fetching an entire record, the relative cost of this fetch decreases by a factor of 5.

What we have shown so far is the number of bits per record used by UjU. Most operations of UjU can be represented using even fewer bits. One can view an operation as a procedure call with parameters. In UjU, every application and every operation within the application has a unique ID that represents a few bits. Apart from these two, the parameters corresponding to a fetch operation is the list of positions with the corresponding search constraint. In practice, we find that most fetch and update operations can be represented in fewer than 100 bits.

In summary, semantic compression can achieve high compression ratios and provide high levels of operator and record level aggregation for a single SMS message.

11.5 Real-world Applications

To demonstrate the real-world effectiveness of UjU, we have implemented and deployed applications for several different organizations. Our UjU field deployments span several contexts and solve different problems in the field for practitioners. We present the details of our field deployments and feedback on the utility of UjU in four different scenarios: i) a pharmacovigilance system in Ghana, ii) a mobile microfinance application deployed in Mexico, iii) a drug list query application for doctors, pharmacists, and health workers, and iv) a customer service application for microfinance customers in Mexico.

11.5.1 Pharmacovigilance System

In Ghana, we worked with Korle-Bu Teaching Hospital to build a Pharmacovigilance system. The goal of this system was twofold: i) track the side effects of a drug among patients, ii) remind patients to take their medication on a timely basis. We briefly describe them below.

First, we divided around 100 drugs into 15 categories. For each category, we record the patient information, medication information such as dosage and schedule. Patients fill out the form (on the phone) and update the database with their information. Doctors then fetch the records stored on the database according to the medication or side-effect.

The second application was a patient reminder system in Ghana that sends out a SMS text message to remind patients to take their medication according to the prescription. As dosage varies for each prescription, the patients needed to be reminded on a timely basis. Our system automatically generates the text message according to the prescription so that the patients are aware of both the time and medication that they need to consume.

We are currently working with Tigo Mobile, a mobile operator, to distribute the application on a larger scale. Tigo Mobile has volunteered to provide SIM cards to patients in this pilot study.

11.5.2 Drug List

Ghanaian doctors and health workers use a book titled Ghana Essential Medicines List that contains a list of commonly used drugs with usage, dosage information, and the level of care required for a particular ailment or disease. Pharmacists check the usage of the medication on the prescription (by referring to Ghana Essential Medicines List) before they give the drug to patients. Our goal was to build a SMS based system that acts as a simple lookup service to Ghana Essential Medicines List and that is easy to use for pharmacists.

Ghana Essential Medicines List contains 29 different categories, of which, some categories contain only few drugs. So, we divided the list into 14 categories and bundled up the rest of the (15) categories into a separate category. The system was designed to be a fetch-only application. The information in the database is not modified by the clients. Beside the information list in Ghana Essential Medicines List, we also store the potential side effects corresponding to each drug. Thus, patient can easy fetch the drug information and check for symptoms of the side effects of the drug. This system has been developed for Korle-Bu Hospital.

11.5.3 Microfinance

CAME is a microfinance institution in Mexico City that primarily provides loans in rural areas [11]. CAME operates based on the concept of group lending. Group meetings, where a group of people living in the nearby village gather on a weekly basis, are held to either accept or return loans.

We developed an application for CAME to support the group meeting and help the CAME officers to judge the credibility of the loanee. To make this assessment the CAME officer needs the credit information of the loanee for the past 16 weeks. Prior to our intervention the CAME officers relied on GPRS to fetch this information from the CAME servers. Our application stores and fetches the client information locally from the mobile device and does not communicate with a central server. The officer stores all the necessary information pertaining to a group on the mobile device leaves to the group meeting. This way, the client information can be fetched locally without receiving any data from a central server, which helps in reducing the data costs.

We also implemented two additional additional features for this application. The first set of features allow remote updates of credit information. The operations for remote updates are: i) Fetching client information from server, ii) Updating the payment information to server for each client, iii) Creating a new record of Interciclo Loan, CAME credit.[1]

The second set of operations allow users to check the loan or balance of each client and updating their records on the server. The following set of *Closing the Cycle* procedures are executed at the last week of the cycle: i) Checks the status of loan of each client that all clients clean up their loan; ii) Distributes the earning from the loan to each client; iii) Renews the loan amount for the next cycle; iv) Allows clients to deposit or withdraw from the savings account; v) Updates the client information to the server.

Compared to the current system that is running on the Palm PDA, CAME officers found our application easy to use due to its simple user interface and the use of keypad on the low end mobile device. One officer felt more comfortable typing the amount from the mobile keyboard than using the touch screen. In the future, we plan to add the functionality of printing the receipt via a Bluetooth enabled mobile printer.

[1]Interciclo is the loan amount in a cycle, where each cycle is of 16 weeks.

11.5.4 Customer Service

Most clients of CAME do not have Internet access. The only way they can communicate with CAME is by making a phone call that costs one peso per minute (0.09). Customers typically wait a long time to talk to the next available customer service representative.

To improve the quality of customer service, we developed a system based on the SMS gateway in UjU. The system can i) record client feedback in the database, ii) update client information, iii) return requested information to the client. With UjU, a client's phone number becomes their primary means of contact and CAME can send information to their clients using SMS.

11.6 Related Work

SMS has been used as a transport for several mobile health applications. FrontlineSMS [21] has been used as a platform for building SMS applications in more than 20 poor countries. The popularity of *FrontlineSMS* can be attributed to its ease of installation and use as well as its flexibility. Besides being used a data center, FrontlineSMS can be an SMS HUB to retransmit messages to a customized group. RapidSMS [53] is another widely used open-source platform that provides a environment for collecting data from a mobile device via SMS Text Message. Successful mobile health applications such as child count [54] have been built on the top of RapidSMS.

OpenXcode [50] is a free open-source software solution for collecting and management any kind of form based data on a mobile device. It allows users to fill out a form and send the information to a central control center from a mobile device. JavaRosa [31] is an open-source platform for collecting form base data as well. It supports XForms standard to create customized form for the user. Project GATHER [22] and CommCare [15] are built on the top of it. GATHER has developed a set of tools that can collect data from different devices. CommCare aims to provide community health workers (CHWs) a platform that support home-based care and social support to HIV, tuberculosis and other chronic patients. OpenMRS is an open-source application that provides an extensible mobile medical record system. It has been adopted by Millennium Villages [42], FACES [19] and AMPATH [7]. Voxiva [68] is a widely used commercial mHealth solution. EpiSurveyor [17] allows easy construction of phone-based data collection systems, but SMS data transmission is in simple plaintext. mPedigree [46] is an ongoing effort geared specifically towards combating the problem of counterfeit drugs in the poor countries.

UjU is intended as a replacement for form-based application platforms to allow easy creation of data collection applications and medical record systems.

11.7 Other Applications

There are a multitude of other possible applications involving structured records that may be implemented on top of the UjU stack. One such application is drug tracking and validation.

According to estimates from the International Chamber of Commerce [78], counterfeit goods constituted 5-7% of the multi-billion dollar pharmaceutical trade in 1997. While this is a problem in rich countries, it is endemic in emerging regions, with the counterfeit rate of certain high-volume drugs reaching up to 80% [79, 165, 246], which poses a clear and present danger to lives and livelihoods.

Conventional strategies to fight counterfeiting include holograms, special packaging, and paper invoice tracing, but each of these have been proven ineffectual in the face of increasingly sophisticated counterfeiting rings, which inject fake drugs into the market for profit and/or sell off genuine medications on the black market or in adjacent countries at marked up prices. Track & trace systems are essentially nonexistent in emerging regions, as the network infrastructure present is not adequate to run systems used in the developed world. As such, the typical methodology relies heavily on paper invoices and signatures, both of which are easily falsifiable.

To fill this gap, we implemented a system called Epothecary which uses 2D barcodes affixed to pharmaceuticals to provide tracking and pedigree authentication. Epothecary uses commodity cameraphones and existing cellular infrastructure. Epothecary is designed to provide end users, retailers, and middle distributors with a reasonable guarantee of drug authenticity by recording all transactions involving medications. Should counterfeit medication be introduced into the system, it provides a mechanism for tracing possible points of entry.

Epothecary's 2D barcodes, or 'tags,' are printed on or affixed to pharmaceuticals at each level of aggregation: each retail package, carton, crate, pallet, etc. Each tag contains a unique, random 20-digit serial number, lot code, and expiration date, and back-end servers track the hierarchy of these serial numbers, e.g. which cartons are contained in which crate, as well as who is in possession of any given unit at any given time all the way down the supply chain. Each participant in the supply chain also receives a unique tag to identify him as a transacting party.

We implemented the client software which runs on the inexpensive Nokia 3110c cameraphone. We use the cameraphone to scan the 2D barcodes of merchandise and Epothecary users and convey compressed, encrypted transaction information over SMS to a central server. Centralized Track & Trace algorithms, tag printing, labeling, scanning, and various transaction types are typical to standard Track & Trace systems.

The transmission of these transactions require reliable SMS delivery which is provided as a part of the UjU stack. Optionally, UjU's semantic compression layer may be applied to the structured metadata of drug units. Epothecary's server logic for managing Track & Trace and barcode verification would be independent of UjU, but UjU's application specification interface could facilitate easy specification of tag formats.

11.8 Chapter Summary

In this chapter we introduced a system called UjU as a platform for developing SMS-based applications. UjU is designed for applications that work with structured data to reduce both initial and operational costs of SMS-based applications. We showed how our system reduces the time to create new aps, achieves high compression levels, and has a low overhead for providing reliability. We designed UjU to require minimal programmer expertise, and showed how UjU applies in three real world applications in Ghana and Mexico. Finally, we showed how parts of the UjU stack may be incorporated as a part of other systems with slightly different requirements than data collection or digitization applications.

Chapter 12

Conclusion and Future Work

My thesis is that web and mobile information access in emerging regions may be enabled through appropriate re-architecting. This dissertation supports this thesis with several contributions in the form of designs, techniques, and systems for enabling information access in emerging regions. Our designs are appropriate to the actual target context while our techniques draw from, and are generalizable to, broader areas of computer science. In total, our systems have been deployed in four countries across three continents. Our projects have had an impact on thousands of people who would otherwise not have access to the information they need.

Web Access - This thesis began by studying web access in emerging regions. We developed ELF, a data collection tool and web optimization engine for emerging regions. ELF simplifies large scale studies of web use by avoiding expertise and security constraints that are common in emerging regions. We used our tool to study and optimize web use in a school in peri-urban India using techniques such as stale page caching, client-side prefetching, and offline browsing. Compared with prior results, we found that despite the constrained context many information needs were similar to those in previous works. In fact, images and video dominated the bandwidth use in our study despite their exceptionally poor performance. The caching and prefetching optimizations we employed were generally successful, yielding a general speedup of 2.2x across all web pages.

To gain a better understanding of how users cope with under-provisioned networks we studied users in an extremely constrained network environment at a large university in rural India. We found chronic frustration with the slow and sometimes intermittent web experience. These severely impaired connections suffering from pathological sharing result in severe networking issues. Our analysis of these shared networks revealed underlying problems with conventional TCP congestion control protocols. The specific issues we observed included: flow unfairness, unpredictability, and recurring timeouts. We defined this region where TCP collapse occurs as the sub-packet regime. From our analysis we developed a model for the sub-packet regime that we then used to develop low bandwidth transport optimizations to avoid TCP collapse in sub-packet regimes.

To address these and other challenges caused by low-bandwidth or high latency connections, we designed RuralCafe, a system that attempts to maximize the utility of data transferred across the network link. RuralCafe explicitly desynchronizes system interactivity from the network state and allows users to continue to make progress. RuralCafe attempts to provide users with information and control over how their network resources are used. In a detailed user study we compared the asynchronous queueing model of RuralCafe against conventional web. We found found that user efficiency with asynchronous queueing is on par with the conventional web interaction model, and has the potential to improve as the connection worsens.

In the worst network conditions, the Internet is only intermittently available or mostly disconnected. To enable digital information access for these areas we designed and implemented a system for automatically generating offline repositories of web content. To generate our CIPs, we designed a custom feature extractor that improves performance for web classification tasks over the full bag of words feature set. This feature extractor is then used by a statistical classifier in our custom focused crawler to gather appropriate content. CIPs may be tailored to any context given a set of desired topics. We used our system to build CIPs for an after school program in India and five senior secondary schools in Kenya that were equipped with computers, but had no Internet access.

We then extended the CIP techniques for caching and prefetching to introduce the concept of vertical caching. We showed that organizing caches based on topics rather than solely URLs can improve the utility of existing cache contents. We then extended this vertical caching concept to prefetching as well.

Finally, after exploring web access in various environments, we found that while our designs were appropriate for each context, many techniques deviated from existing standards. This was the case across environments and throughout the networking stack. We incorporated the lessons learned from our experiences to design a complete architecture for web access in emerging regions. Our system is the first attempt at a comprehensive solution to enabling web access in emerging regions. Many of the techniques we developed are similarly applicable to web access over mobile phones.

Mobile Information Access - In the last two chapters of this thesis we shifted our focus from web access to information through mobile phones. It is generally believed that the high mobile phone penetration presents significant opportunities for profit and development. SMS is particularly popular due to its ubiquity and simplicity, but message costs are high. We showed that relatively powerful applications may be implemented despite the extreme constraints imposed by SMS. Our techniques are generalizable to other environments where network resources are constrained.

SMSFind allows people to search for content on the web using SMS queries for arbitrary topics without a human in the loop. SMSFind leverages and extends techniques from the IR community toward the problem of constrained mobile search across a large, unknown, and noisy document corpus. To design and improve SMSFind, we worked with low income slum inhabitants in Haruma slum before releasing it in a public beta in Nairobi, Kenya.

Where SMSFind is intended for end-users, organizations in emerging regions require robust applications to support their projects. The expertise needed to implement these applications is a scarce and expensive resource. To address this need, we designed the UjU platform to allow non-programmers to quickly and easily develop efficient and reliable SMS-based applications. We used UjU to implement four different SMS-based applications for: (i) a pharmacovigilance system in Ghana, (ii) a mobile microfinance application deployed in Mexico, (iii) a drug list query application for doctors, pharmacists, and health workers in Ghana, and (iv) a customer service application for microfinance customers in Mexico.

12.1 Future Work

This thesis has focused on building appropriate systems for highly constrained environments. This is a new and rich research area with a multitude of different directions for future research. As a result, our discussion here will not be comprehensive. We broadly outline what we perceive to be some of the main challenges in relation to our work.

Understanding User Behavior - In general, it is crucial to understand how people use technology when attempting to design better systems, and our work so far should encourage further research on improving web and mobile access for emerging regions. Our research detailed a few data points for constrained connectivity scenarios, and we plan to explore other contexts to see which results regarding user behavior behind constrained network links are generalizable. Extending our web architecture ideas to mobile web to compare and contrast user behavior is one interesting avenue of research for the future.

New Web Architectures for Emerging Regions - We developed through extensive research what we believe to be a compelling architecture for enabling web access in emerging regions. However, we are fully aware that our design is not the only possible one. There still remain many open questions from an architectural standpoint regarding the design space. Even within each layer of our proposed architecture, specific modifications are open to further consideration. We hope that our work and holistic approach also encourage other researchers to explore the space.

Weighing End-to-end against Middlebox Approaches for Sub-packet Regimes - Specific to the TCP flow collapse problem, our solution was a middle-box approach. In contrast, to solve performance problems at the application layer, we designed ELF as a strictly client-end approach. We welcome future work exploring the tension between these two directions across the spectrum of networking layers and deployment configurations. For example, to avoid TCP collapse, new congestion control protocols at the endpoints that have better pacing another possible alternative.

Designing Interfaces for Asynchronous Web Interactions - Our architecture introduces an asynchronous web browsing model that moves toward disentangling web access from network availability. This paradigm shift affects broad areas of computer science and requires new or modified supporting mechanisms. One major area that we expect future work in is the introduction of new designs by Human Computer Interaction (HCI) researchers to allow a seamless experience despite a gradation of underlying quality of service.

Expanding Vertical Caching and Prefetching - One major component of our architecture that supports the asynchronous web browsing model is our vertical caching and prefetching layer. In this thesis, we introduced a new way to do web caching and prefetching by exploiting the ability of our architecture to allow direct user interaction with cache, and the inherent information aliasing that exists on the web. There is extensive literature on caching, focused crawling, and prefetching to leverage toward improving our new web caching model. We believe that the incorporation of new machine learning and collaborative caching techniques are particularly promising.

Exploring Cost Aware Network Protocols - For the contexts we study and the access models they entail, we find that the appropriateness of conventional performance metrics (e.g. cache hit rate) is questionable. Latency and cost are generally more important considerations in these contexts, and user-centric metrics should be considered in conjunction with low level systems metrics. There is a clear case for this both in the mobile data market where pricing plans are increasingly pay-per-use, as well as for wired connectivity in emerging regions. The success of simplistic approaches such as web content adaptation also suggest that this is a worthwhile direction.

Making Mobile Web Faster - Many of the challenges facing constrained web access are similar for mobile web. Consequently, in our research on enabling mobile information access we only studied problems where the challenges are different. It is apparent from the existing literature that web search over mobile devices is fundamentally different from desktop search. There are entire research fields dedicated to information retrieval problems similar to ours, and undoubtedly many specific techniques will be designed in the future to improve the quality of SMS-based and mobile search. In the future, we are specifically interested in looking for better methods to achieve synergy between new statistical approaches and powerful, but computationally intensive NLP techniques. These advances may not necessarily all be algorithmic, but could also be products of overall system design.

SMS is unique in that it is simultaneously one of the most constrained and ubiquitous communications channels today. In general, SMS or at least bandwidth constrained applications will be around for a long time. As a result, we believe that future work in the area of building systems for SMS is interesting for both intellectual and practical reasons. Future work will hopefully explore how our application stack should be generalized to other classes of applications and pushing the limits of system design over such constrained channels.

Exploring Appropriate Design - Over the past decade there have been many proposed and increasingly numerous deployed projects for improving information access in emerging regions. However, successes are few and modest at best due to the lack of context appropriateness. There are a myriad of social and cultural considerations that are too often neglected; this is particularly true of projects with a technical emphasis. Systems may be too complex to use, require expertise to setup and maintain, or simply not needed by the target population. We hope that future systems and networking research in this space considers these issues and incorporates ideas from other areas within and outside computer science.

Leveraging ICTs for Education - Non-profits, governments, and private entities are increasing spending to equip schools and government offices with computers and, increasingly, Internet connectivity. However, despite the growing emphasis on investing in ICTs, relatively little is known about their actual use and the information needs of these populations. As a result, the investments lack appropriateness or any discernable impact. This is an area of future work that is important to both technology and education research. In Ghana we are working with local educators and philanthropists on a large scale randomized controlled trial to focus on measuring student impact. We hope to use the results from our RCT to show how CIPs may improve school computer labs dramatically with comparatively low budget overheads.

Assessing Impact and Increasing Scale - We assess impact in much of our work, but only at a relatively small scale. While this in itself has been challenging, moving beyond pilots is the next step. We believe that for research in this area to be worthwhile, positive impact and scale should be the eventual metrics for success in future work. Our own efforts in this direction include extending CIPs to more schools in Kenya, and developing SMS-based applications for use by more people in rural areas. Eventually, we hope our works become self-sustainable, if not profitable, ventures, and enough local human capital will accrue over time to make development projects a concept of the past.

Bibliography

[1] http://www.pcmag.com/article2/0,2817,2387318,00.asp.

[2] Adblock plus. https://addons.mozilla.org/en-US/firefox/addon/1865/.

[3] African undersea cables. http://manypossibilities.net/african-undersea-cables/.

[4] Akamai. http://www.akamai.com/.

[5] Akamai broadband adoption trends. http://www.akamai.com/html/technology/dataviz5.html.

[6] Akamai: State of the internet. http://www.akamai.com/stateoftheinternet/.

[7] Ampath. http://medicine.iupui.edu/kenya/hiv.aids.html.

[8] Apache - solr. http://lucene.apache.org/solr/.

[9] Bbc news: Facebook strips down to lite site. http://news.bbc.co.uk/2/hi/technology/8249835.stm.

[10] The bow toolkit. http://www.cs.cmu.edu/~mccallum/bow/.

[11] CAME: Dinero con sentido. `http://www.came.org.mx`.

[12] Carrot2 - open source search results clustering engine. `http://project.carrot2.org/`.

[13] Chacha. `http://www.chacha.com`.

[14] Chacha categories. `http://www.chacha.com/categories`.

[15] Commcare. `http://www.dimagi.com/content/commcare.html`.

[16] echoupal. `http://www.itcportal.com/rural-development/echoupal.htm`.

[17] Episurveyor. `http://www.datadyne.org/episurveyor/`.

[18] Esoko. `http://www.esoko.com/`.

[19] Faces. `http://www.faces-kenya.org/`.

[20] Fasterfox. `http://fasterfox.mozdev.org/`.

[21] Frontlinesms. `http://www.frontlinesms.com/`.

[22] Gather. `http://www.gatherdata.org/`.

[23] Gcash. `http://www.gcash.com/`.

[24] Global - key telecoms, mobile and broadband statistics. `http://www.budde.com.au/`.

[25] Google baraza. `http://www.google.com/baraza/`.

[26] Google sms. `http://www.google.com/sms`.

[27] Google web accelerator. `http://webaccelerator.google.com`.

[28] How many pages in google? take a guess. http://www.nytimes.com/2005/09/27/technology/27search.html.

[29] International telecommunication union. http://www.gsmworld.com/index.htm.

[30] International telecommunication union. http://www.itu.int/.

[31] Javarosa. http://code.javarosa.org.

[32] Just dial. http://www.justdial.com/.

[33] Kannel. http://www.kannel.org/.

[34] Khan academy. http://www.khanacademy.org.

[35] Let's make the web faster - google code. http://code.google.com/speed/articles/web-metrics.html.

[36] Linguistic data consortium. http://www.ldc.upenn.edu.

[37] Loband. http://www.loband.org.

[38] Longwell - simile. http://simile.mit.edu/wiki/Longwell.

[39] Lucene. http://lucene.apache.org.

[40] M-pesa. http://www.safaricom.co.ke/index.php?id=745.

[41] mcheck. http://main.mchek.com/.

[42] Millennium villages. http://www.millenniumvillages.org/.

[43] Mit opencourseware. http://ocw.mit.edu.

[44] Mobile health summit. http://mhealthsummit.org/.

[45] Mobithinking global mobile statistics 2011. http://mobithinking.com/mobile-marketing-tools/latest-mobile-stats/.

[46] Mpedigree. http://www.mpedigree.org/home/.

[47] Ongoing Study in Debre Birhan University, Ethiopia.

[48] Openmrs. http://openmrs.org/wiki/OpenMRS.

[49] Openrosa. http://www.openrosa.org/.

[50] openxcode. http://www.openxdata.org/Main/WebHome.

[51] Opera mini. http://www.opera.com/mobile/download/.

[52] Pump up the volume: An assessment of voice-enabled web search on the iphone. http://www.mcubedigital.com/msearchgroove/.

[53] Rapidsms. http://www.rapidsms.org/.

[54] Rapidsms case study. http://www.rapidsms.org/case-studies/.

[55] Rural bpo. http://www.icmrindia.org/casestudies/catalogue/Business\%20Reports/BREP032.htm.

[56] Sctp. http://www.sctp.org/.

[57] Sms gupshup. http://www.smsgupshup.com/.

[58] Spdy. http://www.chromium.org/spdy/spdy-whitepaper.

[59] Sst. http://pdos.csail.mit.edu/uia/sst/.

[60] State of the mobile web (2011). http://www.opera.com/smw/.

[61] Text retrieval conference. http://trec.nist.gov/.

[62] The tiered-pricing dilemma on smartphones: Overage charges or throttling? http://www.fiercewireless.com/story/tiered-pricing-dilemma-smartphones-overage-charges-or-throttling/2011-05-25.

[63] Trec question answering track. http://trec.nist.gov/data/qamain.html.

[64] Twitter. http://www.twitter.com/.

[65] Unaids. http://www.unaids.org/.

[66] United nations millenium development goals. http://www.un.org/millenniumgoals/.

[67] United villages. http://www.unitedvillages.com.

[68] Voxiva. http://www.voxiva.com/platform.php.

[69] Web initiative for surgical education. http://wise-md.med.nyu.edu.

[70] Websiteoptimization.com. http://www.websiteoptimization.com.

[71] Who. http://www.who.int/countries/mwi/en/.

[72] Wimax forum. http://www.wimaxforum.org.

[73] Windows live mobile. http://home.mobile.live.com/.

[74] World health organization - blue trunk libraries. http://www.who.int/ghl/mobile_libraries/bluetrunk/en.

[75] The world in 2010: Ict facts and figures. http://www.itu.int/ITU-D/ict/material/FactsFigures2010.pdf.

[76] Wwwoffle: World wide web offline explorer. `http://www.gedanken.demon.co.uk/wwwoffle/`.

[77] Yahoo one search. `http://mobile.yahoo.com/onesearch`.

[78] *Countering counterfeiting: A guide to protecting and enforcing intellectual property rights*. ICC Publishing, Paris, France, 1997.

[79] NAFDAC Nigeria - Global Trends. `http://www.nafdacnigeria.org/globaltrends.htm`, 2007.

[80] A. Abouzeid, S. Roy, and M. Azizoglu. Stochastic modeling of tcp over lossy links. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3. IEEE, 2000.

[81] J. Allan. Million query track 2007 overview. Technical report, DTIC Document, 2007.

[82] M. Allman, K. Avrachenkov, U. Ayesta, J. Blanton, and P. Hurtig. Early Retransmit for TCP and SCTP, Jan. 2010. draft-ietf-tcpm-early-rexmt-04 (IETF work in progress).

[83] M. Allman, V. Paxson, and E. Blanton. TCP Congestion Control, Sept. 2009. RFC 5681.

[84] G. Almpanidis, C. Kotropoulos, and I. Pitas. Combining text and link analysis for focused crawling-an application for vertical search engines. *Information Systems*, 2007.

[85] E. Altman, K. Avrachenkov, and C. Barakat. A stochastic model of tcp/ip with stationary random losses. *IEEE/ACM Transactions on Networking*, 13(2), 2005.

[86] S. Amershi and M. Morris. CoSearch: a system for co-located collaborative web search. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2008.

[87] L. L. H. Andrew, S. V. Hanly, and R. G. Mukhtar. Clamp: Active queue management at wireless access points. *European Wireless*, 2005.

[88] R. Angelova and G. Weikum. Graph-based text classification: learn from your neighbors. *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2006.

[89] C. Aone, M. Okurowski, and J. Gorlinsky. Trainable, scalable summarization using robust NLP and machine learning. *Proceedings of the 17th International Conference on Computational linguistics-Volume 1*, 1998.

[90] A. Badam, K. Park, V. Pai, and L. Peterson. Hashcache: Cache storage for the next billion. In *Proceedings of the Sixth USENIX symposium on Networked Systems Design and Implementation*. USENIX Association, 2009.

[91] A. Balasubramanian, Y. Zhou, W. Croft, B. Levine, and A. Venkataramani. Web search from a bus. In *Proceedings of the Second ACM Workshop on Challenged Networks*. ACM, 2007.

[92] E. Baykan, M. Henzinger, L. Marian, and I. Weber. Purely url-based topic classification. *Proceedings of the 18th International Conference on World Wide Web*, 2009.

[93] D. Bergmark, C. Lagoze, and A. Sbityakov. Focused crawls, tunneling, and digital libraries. *Lecture notes in computer science*, 2002.

[94] K. Bharat and M. R. Henzinger. Improved algorithms for topic distillation in a hyperlinked environment. *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1998.

[95] L. Brakmo and L. Peterson. TCP Vegas: End to end congestion avoidance on a global Internet. *IEEE Journal on Selected Areas in Communications*, 13(8), Oct. 1995.

[96] T. Bray. Measuring the web. *Proceedings of the Fifth International World Wide Web Conference*, 1996.

[97] L. Breslau, P. Cao, L. Fan, G. Phillips, and S. Shenker. Web caching and Zipf-like distributions: Evidence and implications. *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, 2002.

[98] E. Brewer, M. Demmer, M. Ho, R. Honicky, J. Pal, M. Plauche, and S. Surana. The challenges of technology research for developing regions. *Pervasive Computing, IEEE*, 5(2), 2006.

[99] E. Brill, S. Dumais, and M. Banko. An analysis of the AskMSR question-answering system. *Proceedings of the ACL-02 Conference on Empirical methods in natural language processing-Volume 10*, 2002.

[100] P. Bruza, R. McArthur, and S. Dennis. Interactive Internet search: keyword, directory and query reformulation mechanisms compared. *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

[101] O. Buyukkokten, H. Garcia-Molina, A. Paepcke, and T. Winograd. Power browser: efficient web browsing for PDAs. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2000.

[102] N. Cardwell, S. Savage, and T. Anderson. Modeling tcp latency. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 3. IEEE, 2000.

[103] D. Carmel, E. Yom-Tov, A. Darlow, and D. Pelleg. What makes a query difficult? *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2006.

[104] S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *ACM SIGMOD Record*, volume 27. ACM, 1998.

[105] S. Chakrabarti, M. van den Berg, and B. Dom. Focused crawling: a new approach to topic-specific web resource discovery. *Computer Networks*, 1999.

[106] J. Charzinski. Traffic Properties, Client Side Cachability and CDN Usage of Popular Web Sites. *Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance*, 2010.

[107] G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical report, TR2000-381, Department of Computer Science, Dartmouth College, 2000.

[108] J. Chen, S. Amershi, A. Dhananjay, and L. Subramanian. Comparing web interaction models in developing regions. *Proceedings of the First ACM Symposium on Computing for Development*, 2010.

[109] J. Chen, D. Hutchful, W. Thies, and L. Subramanian. Analyzing and accelerating web access in a school in peri-urban india. *Proceedings of the 20th International Conference companion on World Wide Web*, 2011.

[110] J. Chen, T. Karthik, and L. Subramanian. Contextual information portals. *Proceedings of AAAI Spring Symposium*, 2010.

[111] J. Chen, B. Linn, and L. Subramanian. Sms-based contextual web search. *Proceedings of the First ACM Workshop on Networking, Systems, and Applications for Mobile Handhelds*, 2009.

[112] J. Chen, R. Power, L. Subramanian, and J. Ledlie. Design and implementation of contextual information portals. *Proceedings of the 20th International Conference companion on World Wide Web*, 2011.

[113] J. Chen, L. Subramanian, and E. Brewer. Sms-based web search for low-end mobile devices. *Proceedings of the International Conference on Mobile Computing and Networking*, 2010.

[114] J. Chen, L. Subramanian, and J. Li. Ruralcafe: web search in the rural developing world. *Proceedings of the 18th International Conference on World Wide Web*, 2009.

[115] J. Chen, L. Subramanian, and K. Toyama. Web search and browsing behavior under poor connectivity. *Proceedings of the 27th International Conference on Human Factors in Computing Systems*, 2009.

[116] J. Cho, H. Garcia-Molina, and L. Page. Efficient crawling through url ordering. *Proceedings of the Seventh International Conference on World Wide Web*, 1998.

[117] C. Clarke, G. Cormack, G. Kemkes, M. Laszlo, T. Lynam, E. Terra, and P. Tilker. Statistical selection of exact answers (MultiText experiments for TREC 2002). *Proceedings of TREC*, 2002.

[118] C. Clarke, N. Craswell, and I. Soboroff. Overview of the TREC 2004 terabyte track. *Proceedings of TREC*, 2004.

[119] Critical Mass: The Worldwide State of the Mobile Web. The Nielsen Company. 2008.

[120] H. Dang, D. Kelly, and J. Lin. Overview of the TREC 2007 question answering track. *Proceedings of TREC*, 2007.

[121] B. D. Davison. Topical locality in the web. *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

[122] M. Demmer, B. Du, and E. Brewer. Tierstore: a distributed filesystem for challenged networks in developing regions. In *Proceedings of the Sixth USENIX Conference on File and Storage Technologies*. USENIX Association, 2008.

[123] J. Domčnech, J. Gil, J. Sahuquillo, and A. Pont. Web prefetching performance metrics: A survey. *Performance Evaluation*, 63(9-10), 2006.

[124] J. Domčnech, A. Pont, J. Sahuquillo, and J. Gil. A user-focused evaluation of web prefetching algorithms. *Computer Communications Review*, 30(10), 2007.

[125] B. Du, M. Demmer, and E. Brewer. Analysis of WWW traffic in Cambodia and Ghana. *Proceedings of the 15th International Conference on World Wide Web*, 2006.

[126] D. Duchamp. Prefetching hyperlinks. *Proceedings of the Second Conference on USENIX Symposium on Internet Technologies and Systems-Volume 2*, 1999.

[127] S. Dumais and H. Chen. Hierarchical classification of Web content. *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2000.

[128] M. Ehrig and A. Maedche. Ontology-focused crawling of Web documents. *Proceedings of the 2003 ACM symposium on Applied computing*, 2003.

[129] K. Fall. A delay-tolerant network architecture for challenged internets. In *Proceedings of the 2003 Conference on Applications, technologies, architectures, and protocols for computer communications*. ACM, 2003.

[130] L. Fan, P. Cao, W. Lin, and Q. Jacobson. Web prefetching between low-bandwidth clients and proxies: potential and performance. *Proceedings of the ACM SIGMETRICS International Conference on Measurement and modeling of computer systems*, 1999.

[131] C. Fellbaum. *WordNet: An electronic lexical database*. MIT press Cambridge, MA, 1998.

[132] R. Fielding, J. Gettys, J. Mogul, H. Frystyk, L. Masinter, P. Leach, and T. Berners-Lee. RFC 2616: Hypertext transfer protocol–HTTP/1.1, June 1999. *Status: Standards Track*.

[133] L. Finkelstein, E. Gabrilovich, Y. Matias, E. Rivlin, Z. Solan, G. Wolfman, and E. Ruppin. Placing search in context: The concept revisited. *ACM Transactions on Information Systems*, 20(1), 2002.

[134] S. Floyd and T. Henderson. The NewReno Modification to TCP's Fast Recovery Algorithm, Apr. 1999. RFC 2582.

[135] S. Floyd and V. Jacobson. Random early detection gateways for congestion avoidance. *IEEE/ACM Transactions on Networking*, 1(4), Aug. 1993.

[136] S. Fortin-Parisi and B. Sericola. A Markov model of TCP throughput, goodput and slow start. *Performance Evaluation*, 58(2-3), 2004. Distributed Systems Performance.

[137] K. Gaikwad, G. Paruthi, and W. Thies. Interactive DVDs as a Platform for Education. *IEEE/ACM International Conference on Information and Communication Technologies and Development*, 2010.

[138] S. Gao, W. Wu, C. Lee, and T. Chua. A maximal figure-of-merit learning approach to text categorization. *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2003.

[139] J. Gil. Modelling TCP with a Discrete Time Markov Chain. *HET-NETs: Performance Modelling and Evaluation of Heterogenous Networks*, July 2005.

[140] H. Guan, J. Zhou, and M. Guo. A class-feature-centroid classifier for text categorization. *Proceedings of the 18th International Conference on World Wide Web*, 2009.

[141] S. Guo, M. Falaki, E. Oliver, S. Rahman, A. Seth, M. Zaharia, and S. Keshav. Very low-cost internet access using KioskNet. *ACM Computer Communications Review*, 2007.

[142] S. Ha, I. Rhee, and L. Xu. Cubic: A new tcp-friendly high-speed tcp variant. *ACM SIGOPS Operating Systems Review*, 42(5), 2008.

[143] M. Handley, S. Floyd, J. Padhye, and J. Widmer. TCP friendly rate control (TFRC): Protocol specification, Jan. 2003. RFC 3448.

[144] D. Harman. Overview of the first text retrieval Conference (TREC-1). *First Text Retrieval Conference (Trec-1): Proceedings*, 1993.

[145] M. Hersovici, M. Jacovi, Y. S. Maarek, D. Pelleg, M. Shtalhaim, and S. Ur. The shark-search algorithm. an application: tailored web site mapping. *Proceedings of the International Conference on World Wide Web*, 1998.

[146] E. Huerta and R. Sandoval-Almazán. Digital literacy: Problems faced by telecenter users in Mexico. *Information Technology for Development*, 13(3), 2007.

[147] B. Hull, V. Bychkovsky, Y. Zhang, K. Chen, M. Goraczko, A. Miu, E. Shih, H. Balakrishnan, and S. Madden. Cartel: a distributed mobile sensor computing system. In *Proceedings of the 4th International Conference on Embedded networked sensor systems*. ACM, 2006.

[148] S. Ihm, K. Park, and V. Pai. Towards Understanding Developing World Traffic. *Proceedings of the ACM Workshop on Networked Systems for Developing Regions*, 2010.

[149] S. Ihm, K. Park, and V. Pai. Wide-area network acceleration for the developing world. In *Proceedings of the 2010 USENIX Conference on USENIX Annual Technical Conference*. USENIX Association, 2010.

[150] S. Isaacman and M. Martonosi. Potential for collaborative caching and prefetching in largely-disconnected villages. *Proceedings of the ACM Workshop on Wireless Networks and Systems for Developing Regions*, 2008.

[151] S. Isaacman and M. Martonosi. The C-LINK System for Collaborative Web Usage: A Real-World Deployment in Rural Nicaragua. *Proceedings of the ACM Workshop on Networked Systems for Developing Regions*, 2008.

[152] S. Isaacman and M. Martonosi. Low Infrastructure Methods to Improve Internet Access for Mobile Users in Emerging Regions. *Proceedings of the 20th International Conference on World Wide Web*, 2011.

[153] R. Jain, D. Chiu, and W. Hawe. A quantitative measure of fairness and discrimination for resource allocation in shared computer systems. Technical report, Sept. 1984. DEC Research Report TR-301.

[154] S. Jain, K. Fall, and R. Patra. Routing in a delay tolerant network. *ACM SIGCOMM Computer Communication Review*, 34(4), 2004.

[155] T. Joachims. Making large scale svm learning practical. *Advances in Kernel Methods, Support Vector Learning*, 1998.

[156] T. Joachims, C. Nedellec, and C. Rouveirol. Text categorization with support vector machines: learning with many relevant. *10th European Conference on Machine Learning*, 1998.

[157] D. Johnson, E. Belding, K. Almeroth, and G. van Stam. Internet usage and performance analysis of a rural wireless network in Macha, Zambia. *Proceedings of the ACM Workshop on Networked Systems for Developing Regions*, 2010.

[158] K. S. Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28, 1972.

[159] M. Kamvar and S. Baluja. A large scale study of wireless search behavior: Google mobile search. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2006.

[160] M. Kamvar, M. Kellar, R. Patel, and Y. Xu. Computers and iphones and mobile phones, oh my!: a logs-based comparison of search users on different devices. *Proceedings of the 18th International Conference on World Wide Web*, 2009.

[161] M. Kan and H. Thi. Fast webpage classification using url features. In *Proceedings of the 14th ACM International Conference on Information and Knowledge Management*. ACM, 2005.

[162] T. Kanungo and D. Orr. Predicting the readability of short web summaries. *Proceedings of the Second ACM International Conference on Web Search and Data Mining*, 2009.

[163] A. Karlson, B. Bederson, and J. SanGiovanni. AppLens and launchTile: two designs for one-handed thumb use on small devices. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2005.

[164] D. Katabi, M. Handley, and C. Rohrs. Congestion control for high bandwidth-delay product networks. In *ACM SIGCOMM Computer Communication Review*, volume 32. ACM, 2002.

[165] T. Kelesidis, I. Kelesidis, P. Rafailidis, and M. Falagas. Counterfeit or substandard antimicrobial drugs: a review of the scientific evidence. *Journal of Antimicrobial Chemotherapy*, 60(2), August 2007.

[166] King, A. B. Website Optimization Secrets. 2008.

[167] J. M. Kleinberg. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, 1999.

[168] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek. The Click modular router. *ACM Transactions on Computer Systems*, 2000.

[169] P. Kotkar, W. Thies, and S. Amarasinghe. An audio wiki for publishing user-generated content in the developing world. *HCI for Community and International Development*, 2008.

[170] R. Kraft, C. Chang, F. Maghoul, and R. Kumar. Searching with context. *Proceedings of the 15th International Conference on World Wide Web*, 2006.

[171] A. Kumar. Comparative Performance Analysis of Versions of TCP in a Local Network with a Lossy Link. *IEEE/ACM Transactions on Networking*, 1998.

[172] A. Kumar, S. Agarwal, and P. Manwani. The spoken web application framework: user generated content and service creation through low-end mobiles. *Proceedings of the 2010 International Cross Disciplinary Conference on Web Accessibility*, 2010.

[173] T. V. Lakshman and U. Madhow. The performance of TCP/IP for networks with high bandwidth-delay products and random loss. *IEEE/ACM Transactions on Networking*, 1997.

[174] H. Lam and P. Baudisch. Summary thumbnails: readable overviews for small screen web browsers. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2005.

[175] S. Lawrence. Context in web search. *IEEE Data Engineering Bulletin*, 23(3), 2000.

[176] D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. *Lecture Notes in Computer Science*, 1998.

[177] X. Li and D. Roth. Learning question classifiers. *Proceedings of the 19th International Conference on Computational linguistics-Volume 1*, 2002.

[178] C. Lin. Training a selection function for extraction. *Proceedings of the Eighth International Conference on Information and Knowledge Management*, 1999.

[179] J. Lin and B. Katz. Question answering techniques for the World Wide Web. *EACL-2003 Tutorial*, 2003.

[180] T. Loon and V. Bharghavan. Alleviating the latency and bandwidth problems in WWW browsing. *Proceedings of USENIX Symposium on Internet Technology and Systems*, 1997.

[181] R. Ludwig and M. Meyer. The Eifel Detection Algorithm for TCP, Apr. 2003. RFC 3522.

[182] R. Luk, M. Ho, and P. Aoki. Asynchronous remote medical consultation for Ghana. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2008.

[183] C. D. Manning, P. Raghavan, and H. Schutze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.

[184] L. Massouli and J. Roberts. Arguments in favour of admission control for tcp flows. *ITC 16: 16th International Teletraffic Congress*, 1999.

[185] M. Mathis, J. Mahdavi, S. Floyd, and A. Romanow. TCP selective acknowledgment options, Oct. 1996. RFC 2018.

[186] M. Mathis, J. Semke, J. Mahdavi, and T. Ott. The macroscopic behavior of the tcp congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review*, 27(3), 1997.

[187] A. K. McCallum. Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering. `http://www.cs.cmu.edu/~mccallum/bow`, 1996.

[188] P. McKenney. Stochastic fairness queueing. In *INFOCOM'90. Ninth Annual Joint Conference of the IEEE Computer and Communication Societies*. IEEE, 1990.

[189] F. Menczer, G. Pant, P. Srinivasan, and M. E. Ruiz. Evaluating topic-driven web crawlers. *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

[190] V. Misra, W.-B. Gong, and D. Towsley. Stochastic differential equation modeling and analysis of tcp-windowsize behavior. *Performance '99*, 1999.

[191] J. Mogul, F. Douglis, A. Feldmann, and B. Krishnamurthy. Potential benefits of delta encoding and data compression for http. *ACM SIGCOMM Computer Communication Review*, 27(4), 1997.

[192] R. Morris. Tcp behavior with many flows. In *Proceedings of the Fifth International Conference on Network Protocols*, volume 97, 1997.

[193] J. Mothe and L. Tanguy. Linguistic features to predict query difficulty-a case study on previous trec campaigns. *SIGIR Workshop on Predicting Query Difficulty-Methods and Applications*, 2005.

[194] S. Mubaraq, J. Hwang, D. Filippini, R. Moazzami, L. Subramanian, and T. Du. Economic analysis of networking technologies for rural developing regions. *Workshop on Internet Economics*, 2005.

[195] M. Najork and J. Wiener. Breadth-first crawling yields high-quality pages. *Proceedings of the 10th International Conference on World Wide Web*, 2001.

[196] K. Nigam, J. Lafferty, and A. McCallum. Using maximum entropy for text classification. In *IJCAI-99 Workshop on Machine Learning for Information Filtering*, volume 1, 1999.

[197] E. Oliver. Exploiting the Short Message Service as a Control Channel in Challenged Network Environments. *Proceedings of the Third ACM Workshop on Challenged Networks*, 2008.

[198] E. Oliver. Characterizing the transport behaviour of the short message service. *MobiSys '10*, 2010.

[199] J. Ott and D. Kutscher. Drive-thru internet: Ieee 802.11 b for automobile users. In *INFOCOM 2004. Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1. IEEE, 2004.

[200] B. Oyelaran-Oyeyinka and C. Nyaki Adeya. Internet access in africa: empirical evidence from kenya and nigeria. *Telematics and Informatics*, 21(1), 2004.

[201] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose. Modeling tcp throughput: A simple model and its empirical validation. In *ACM SIGCOMM Computer Communication Review*, volume 28. ACM, 1998.

[202] V. Padmanabhan and J. Mogul. Using predictive prefetching to improve World Wide Web latency. *ACM SIGCOMM Computer Communication Review*, 26(3), 1996.

[203] V. Pai, A. Badam, S. Ihm, and K. Park. First-class access for developing-world environments. *Proceedings of the Fifth International Conference on Future Internet Technologies*, 2010.

[204] M. Paik, J. Chen, and L. Subramanian. Epothecary - cost effective drug pedigree tracking using mobile phones. *Proceedings of the First ACM Workshop on Networking, Systems, and Applications for Mobile Handhelds*, 2009.

[205] M. Paik et al. The Case for SmartTrack. *IEEE/ACM Conference on Information and Communication Technologies and Development*, 2009.

[206] S. Pandey and C. Olston. Crawl ordering by search impact. In *Proceedings of the International Conference on Web Search and Web Data Mining*. ACM, 2008.

[207] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *Proceedings of the First International Conference on Scalable Information Systems*, 2006.

[208] N. Patel, D. Chittamuru, A. Jain, P. Dave, and T. Parikh. Avaaj Otalo. A Field Study of an Interactive Voice Forum for Small Farmers in Rural India. *Proceedings of the 28th International Conference on Human Factors in Computing Systems*, 2010.

[209] R. Patra, S. Nedevschi, S. Surana, A. Sheth, L. Subramanian, and E. Brewer. Wildnet: Design and implementation of high performance wifi based long distance networks. In *Proceedings of the Fifth USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 2007.

[210] U. Pawar, J. Pal, and K. Toyama. Multiple mice for computers in education in developing countries. *Proceedings of International Conference on Information and Communication Technologies and Development*, 2006.

[211] V. Paxson and M. Allman. Computing TCP's Retransmission Timer, Nov. 2000. RFC 2988.

[212] C. Pazos, J. Sanchez Agrelo, and M. Gerla. Using back-pressure to improve tcp performance with many flows. In *INFOCOM'99. Eighteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 2. IEEE, 1999.

[213] T. Peng, C. Zhang, and W. Zuo. Tunneling enhanced by web page content block partition for focused crawling: Research Articles. *Concurrency and Computation: Practice & Experience*, 2008.

[214] A. Pentland, R. Fletcher, and A. Hasson. DakNet: rethinking connectivity in developing nations. *Computer*, 37(1), 2004.

[215] A. Pentland, R. Fletcher, and A. Hasson. Daknet: Rethinking connectivity in developing nations. *Computer*, 2004.

[216] Port80 Software. `http://www.port80software.com`.

[217] X. Qi and B. Davison. Web page classification: Features and algorithms. *ACM Computing Surveys*, 2009.

[218] J. Qin, Y. Zhou, and M. Chau. Building domain-specific web collections for scientific digital libraries: a meta-search enhanced focused crawling method. *Proceedings of the Fourth ACM/IEEE-CS joint Conference on Digital libraries*, 2004.

[219] L. Qiu, Y. Zhang, and S. Keshav. Understanding the performance of many TCP flows* 1. *Computer Networks*, 37(3-4), 2001.

[220] M. Rabinovich and O. Spatscheck. Web Caching and Replication. *SIGMOD Record*, 32(4), 2003.

[221] A. Ratan, S. Satpathy, L. Zia, K. Toyama, S. Blagsvedt, U. Pawar, T. Subramaniam, and A. Ratan. Kelsa+: Digital Literacy for Low-Income Office Workers. *Proceedings of International Conference on Information and Communication Technologies and Development*, 2009.

[222] A. Reda, E. Cutrell, and B. Noble. Towards improved web acceleration: Leveraging the personal web. *Proceedings of ACM Workshop on Networked Systems for Developing Regions*, 2011.

[223] A. Reda, Q. Duong, T. Alperovich, B. Noble, and Y. Haile. Robit: An Extensible Auction-based Market Platform for Challenged Environments. *Proceedings of the 4th International Conference on Information and Communications Technologies and Development*, 2010.

[224] A. Reda, B. Noble, and Y. Haile. Distributing private data in challenged network environments. *Proceedings of the 19th International Conference on World wide web*, 2010.

[225] S. Rhea, K. Liang, and E. Brewer. Value-based web caching. *Proceedings of the 12th International Conference on World Wide Web*, 2003.

[226] U. Saif, A. Chudhary, S. Butt, N. Butt, and G. Murtaza. Internet for the developing world: Offline internet access at modem-speed dialup connections. *Proceedings of International Conference on Information and Communication Technologies and Development*, 2007.

[227] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav. Low-cost communication for rural internet kiosks using mechanical backhaul. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*. ACM, 2006.

[228] A. Seth, D. Kroeker, M. Zaharia, S. Guo, and S. Keshav. Low-cost communication for rural internet kiosks using mechanical backhaul. In *Proceedings of the 12th Annual International Conference on Mobile Computing and Networking*. ACM, 2006.

[229] H. Shen, M. Kumar, S. Das, and Z. Wang. Energy-efficient data caching and prefetching for mobile devices based on utility. *Mobile Networks and Applications*, 10(4), 2005.

[230] T.-L. Sheu and L.-W. Wu. An analytical model of fast retransmission and recovery in tcp-reno. *IEEE International Conference on Networks*, Nov. 2004.

[231] T.-L. Sheu and L.-W. Wu. An analytical model of fast retransmission and recovery in tcp-sack. *Performance Evaluation*, 64(6), 2007.

[232] T.-L. Sheu and L.-W. Wu. Modeling of multiple tcp segment losses in slow start and congestion avoidance. *Booktitle of Information Science and Engineering*, 2007.

[233] W. Shi, E. Collins, and V. Karamcheti. Modeling object characteristics of dynamic web content. *Journal of Parallel and Distributed Computing*, 2003.

[234] L. Shih and D. Karger. Using urls and table layout for web classification tasks. *Proceedings of the 13th International Conference on World Wide Web*, 2004.

[235] B. Sikdar, S. Kalyanaraman, and K. S. Vastola. Analytic models for the latency and steady-state throughput of tcp tahoe, reno, and sack. *IEEE/ACM Transactions on Networking*, 11(6), 2003.

[236] A. Singhal and M. Kaszkiel. A case study in web search using TREC algorithms. *Proceedings of the 10th International Conference on World Wide Web*, 2001.

[237] T. Sohn, K. Li, W. Griswold, and J. Hollan. A diary study of mobile information needs. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2008.

[238] K. Song, Q. Zhang, and M. Sridharan. Compound tcp: A scalable and tcp-friendly congestion control for high-speed networks. In *INFOCOM 2006. Twenty-fifth AnnualJoint Conference of the IEEE Computer and Communications Societies*, 2006.

[239] M. Soubbotin and S. Soubbotin. Use of patterns for detection of answer strings: A systematic approach. *Proceedings of TREC*, 11, 2002.

[240] N. Spring, M. Chesire, M. Berryman, V. Sahasranaman, T. Anderson, and B. Bershad. Receiver based management of low bandwidth access links. In *INFOCOM 2000. Nineteenth Annual Joint Conference of the IEEE Computer and Communications Societies*, volume 1. IEEE, 2000.

[241] L. Subramanian, S. Nedevschi, R. Patra, S. Surana, A. Sheth, and E. Brewer. Rethinking wireless for the developing world. In *In Proceedings of Fifth Workshop on Hot Topics in Networks*, 2006.

[242] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz. Overqos: An overlay based architecture for enhancing internet qos. In *Proceedings of the First Conference on Symposium on Networked Systems Design and Implementation*. USENIX Association, 2004.

[243] O. Suominen, K. Viljanen, and E. HyvAnen. User-centric faceted search for semantic portals. *The Semantic Web: Research and Applications*, 2007.

[244] S. Surana, R. Patra, S. Nedevschi, M. Ramos, L. Subramanian, Y. Ben-David, and E. Brewer. Beyond pilots: keeping rural wireless networks alive. In *Proceedings of the Fifth USENIX Symposium on Networked Systems Design and Implementation*. USENIX Association, 2008.

[245] W. Thies, J. Prevost, T. Mahtab, G. Cuevas, S. Shakhshir, A. Artola, B. Vo, Y. Litvak, S. Chan, S. Henderson, et al. Searching the world Wide Web in low-connectivity communities. *Proceedings of the 11th International Conference on World Wide Web*, 2002.

[246] R. Tomlinson. China cracks down on counterfeit medicines. *British Medical Journal*, 316(7184), March 6 1999.

[247] R. Wang, S. Sobti, N. Garg, E. Ziskind, J. Lai, and A. Krishnamurthy. Turning the postal system into a generic digital communication mechanism. In *ACM SIGCOMM Computer Communication Review*, volume 34. ACM, 2004.

[248] L. Wei-Chih, M. Tierney, J. Chen, F. Kazi, A. Hubard, J. Pasquel, L. Subramanian, and B. Rao. Uju: Sms-based applications made easy. *Proceedings of the First ACM Symposium on Computing for Development*, 2010.

[249] H. Weinreich, H. Obendorf, E. Herder, and M. Mayer. Off the beaten tracks: exploring three aspects of web navigation. *Proceedings of the 15th International Conference on World Wide Web*, 2006.

[250] A. Woodruff, P. Aoki, E. Brewer, P. Gauthier, and L. Rowe. An investigation of documents from the World Wide Web. *Proceedings of the Fifth International World Wide Web Conference*, 1996.

[251] S. Wyche, T. Smyth, M. Chetty, P. Aoki, and R. Grinter. Deliberate interactions: characterizing technology use in nairobi, kenya. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, 2010.

[252] X. Xie, G. Miao, R. Song, J. Wen, and W. Ma. Efficient browsing of web search results on mobile devices based on block importance model. *Proceedings of Pervasive Computing and Communications*, 2005.

[253] Y. Yang, S. Slattery, and R. Ghani. A study of approaches to hypertext categorization. *Journal of Intelligent Information Systems*, 2002.

[254] A. Yaw, H. Carl, B. Waylon, L. Adam, T. Clint, and B. Gaetano. Open data kit: Building information services for developing regions. *IEEE/ACM Conference on Information and Communication Technologies and Development*, 2010.

[255] J. Yi, F. Maghoul, and J. Pedersen. Deciphering mobile search patterns: a study of yahoo! mobile search queries. *Proceedings of the 17th International Conference on World Wide Web*, 2008.

[256] L. Zhang, S. Michel, K. Nguyen, A. Rosenstein, S. Floyd, and V. Jacobson. Adaptive Web Caching: Towards a New Global Caching Architecture. *Third International Caching Workshop, June*, 1998.

[257] X. Zhang, J. Kurose, B. Levine, D. Towsley, and H. Zhang. Study of a bus-based disruption-tolerant network: mobility modeling and impact on routing. In *Proceedings of the 13th Annual ACM International Conference on Mobile Computing and Networking*. ACM, 2007.

[258] H. Zheng, B. Kong, and H. Kim. Learnable focused crawling based on ontology. *Lecture Notes in Computer Science*, 2008.