**Big Data Paper/Project Notes and References**

**Issues with Browsing over Mobile Wireless Network (High Level):**

* High Cost
* High Latency
* Low Bandwidth
* Unreliability

**Inefficiencies that cause issues (Low Level):**

* Connection overhead
* Redundant transmission
* Verbose protocol

We intercept and control communications over wireless in order to reduce traffic volume and optimize caching in the proxy server and mobile device to reduce data transmission and ultimately reduce bandwidth.

We insert two component into the datapath between the web client and web server – the Client Side Intercept (CSI) and Server Side Intercept (SSI). The CSI implements caching on the client side and SSI implements caching on the server side.

**Strengths:**

* Doesn’t affect browser functionality and independent of web browser and web server.
* What we need: a control simulator.

**Weaknesses:**

* Assumptions decreasing performance: expiration, Most recently used.

We aim to achieve this by implementing a cache on both CSI and SSI.

If the necessary webpage is on CSI, it simply returns it without any network transmissions. If it is not in the CSI, but is found in the SSI, then it returns the page from the proxy. If it is in netiher of these caches, it has to obtain it from the web server The issue with this is that these caching techniqeus do not help in CGI processing where each request returns a different result (for example, a stock-quote server). For these, though the majority of the page is redundant, we would have to reload the entire page from the web server, which puts a strain on both the webserver load as well as network bandwidth. We take advantage of the fact that different replies from the same program are usually very similar.

However, unlike the ordinary http cache used today, we use a differencing method .

Need to address:

How CGI works?

Other CGI?

Cache compressed version or uncompressed version?

Priorities:

1. implement url cache without modifying output of what we have
2. experiments (multiple mobile phones, multiple webpages)
3. rest of the stuff below.

The system is transparent and works with any browser and any phone

Mobile phone browser usage patterns

Other research uses “delta” as a black box and takes a high level look

Request: A-IM: xdelta

Delta-Base: /4iYlzVw…

Response: IM: xdelta

Bit rate of wireless links?

Datarate = link speed/#devices

Q: 10 byte chunks vs. each FP = 4bytes.

**To-do:**

**461:** How do we redirect our browser to go through the proxy?

**Simulation2:** Set user agent. Gzip. Check expiration time. Advantages: improves latency. Tradeoff: space vs. network.

Morning, evening, afternoon.

Take advantage of additional information.

**Check** general cache sizes for mobile and proxy

**For each file:** include url, expiration.

**Tradeoff:** # of bytes vs FP size

**To-do:**

1. Mobile checks cache for url and expiration date
2. Mobile gives proxy the header
3. Proxy checks cache for url and expiration date
4. Proxy saves info in file and calls networked client.
5. Client makes a request to networked server and gets response and stores as file.
6. Proxy then sends uses the information in the response file.
7. Proxy takes the response and chunks and stores it in the cache.
8. Proxy responds to mobile.
9. Get offline data with amazon with client and server proxy and see how it compares to current data.
10. Get offline data with browser and server proxy and see how it compares to current data.

Assumptions: cacheability

Weakness: we don’t do a “check” for freshness.

Include information about CGI

Weakness, overlap inbetween different chunks.

MOBILE

URL in cache?

NO? SEND REQUEST WITH NULL LIST

YES?

URL expired?

NO? USE IT

YES? SEND REQUEST WITH ARRAYLIST

PROXY

URL in cache?

NO? SEND REQUEST

YES?

URL expired?

NO? SEND TO MOBILE WITH DIFF CHUNKS

YES? SEND REQUEST. SEND TO MOBILE WITH DIFF CHUNKS

Proxy sends back: arraylist of all FPS and the chunks associated with the diffs. Mobile then adds the entire arraylist to the url cache and adds the diff to mru cache.

**Strengths**

Mobile doesn’t have to compute FPs – limited computational capacity – tradeoff

Review of literature: delta compression, rsync.. we focus on mobile.

From our progress Report:

Madhu Jayakumar, Marcela Melara, Nayden Nedev

COS598C

Prof. Li

13 March 2013

Big Data Project Report

Mobile devices have limited computational resources available and there is a clear trend of a growing base of mobile device users on a monthly data plan. Our project is to work on decreasing mobile data bandwidth consumption due to mobile web browsing by leveraging the redundancies of websites. This will not only reduce the data consumption for the user from the network provider, but could also decrease server workload, network traffic volume, and latency and increase performance and possibly battery life. Our approach is to find redundancies on a chunk-by-chunk basis, fingerprinting these chunks and using the delta deduplication technique. The goal is to build a software application which will divide incoming HTTP traffic into fixed-sized chunks (approx. 64KB), apply a fingerprinting function (Rabin or something similar), and additionally apply the delta to ultimately find the redundancies between incoming sites, and sites cached on a designated proxy server. By finding these redundancies before the sites are loaded onto the mobile device instead of loading the full sites every time, we can reduce the bandwidth consumption, and thus the amount of data transferred, with the ultimate aim to save service provider data plan costs. We will be working with the three development android phones provided by Kai Li and running the proxy set up by Zhe Wang. Our plan of action for the first few steps is to browse various webpages using these phones and obtain a TCP dump (through use of TCPSniffer and/or Wireshark), which will contain the TCP/IP header as well as payload information. We will then look into analyzing redundancies in the payload transfer using fingerprinting.

We identify two scenarios in which we could find redundancies between sites: (1) Two visits to the same website a specific time-interval apart (e.g. visit CNN.com in the morning and in the evening on the same day), and (2) A visit to two different sites which show very similar content (e.g. two different news sites).

With respect to related work, Qian et al. perform a network-wide study of redudant transfers caused by inefficient caching on smartphones (Web caching on smartphones: ideal vs reality, MobiSys '12). However, they only look at redudant transfers of the content of entire web pages, unlike our work which looks at intra-page redundancies. Other work has focused on characterizing modern web traffic caching as well. For instance, Ihm and Pai study web traffic over several years (Towards understanding modern web traffic caching, SIGMETRICS '11) but do not look at mobile web traffic.