Performance Observations of Fennec Development on Mobile Networks

June 30 2008

mcmanus@ducksong.com

This memo is an interim status report. In particular, in contains very few conclusions and those conclusions it does contain should be considered pre-mature. However it does have some interesting data – some of it confirming what we would expect to see, and some unexpected tidbits. Most importantly, along with the attached spreadsheet, it serves as a baseline for evaluating future changes.

In instances where specific avenues of investigation and/or improvement are identified they are clearly highlighted in ***bold/italic*** and future work items summarized towards the end of the memo.

METHODOLOGY

As noted on the mobile newsgroup, I built a tool that integrates a web server, the fennec client, and a simulated mobile network all on one Linux host. I included copies of seven different websites and eight different network configurations. Each website was identified because it was believed to provide some kind of challenging profile for the browser (usually size of data or number of different objects), and the network configurations reflected a literature review of real life conditions on various popular mobile networks.

Each configuration will be explained after a summary of the data gathered – because graphs are the fun stuff!

EFFICIENCY METRICS EXPLAINED

The primary metric I focus on is the “Efficiency Ratio” of a loaded page. The metric is a number between 0 (inefficient) and 1 (perfectly efficient). To calculate the number, find the ratio of the goodput rate for a total page to the theoretical maximum downstream bandwidth of the network it is running on.

For instance, if it takes 2 seconds to download a web page containing 40KB of HTML and two images of 5KB each that is a “goodput” of 25KB/s. Generally throughputs on wireless networks are measured in kbit/s so our goodput is canonically 200Kbit/s. If the network the test was measured on was advertised as 250kbit/s then we would have an efficiency ratio of .8 (Ice. 200/250). As goodput only measures the “good” stuff, actual HTML and image data, and ignores HTTP, TCP, IP, and L2 overhead - .8 is a pretty good number.

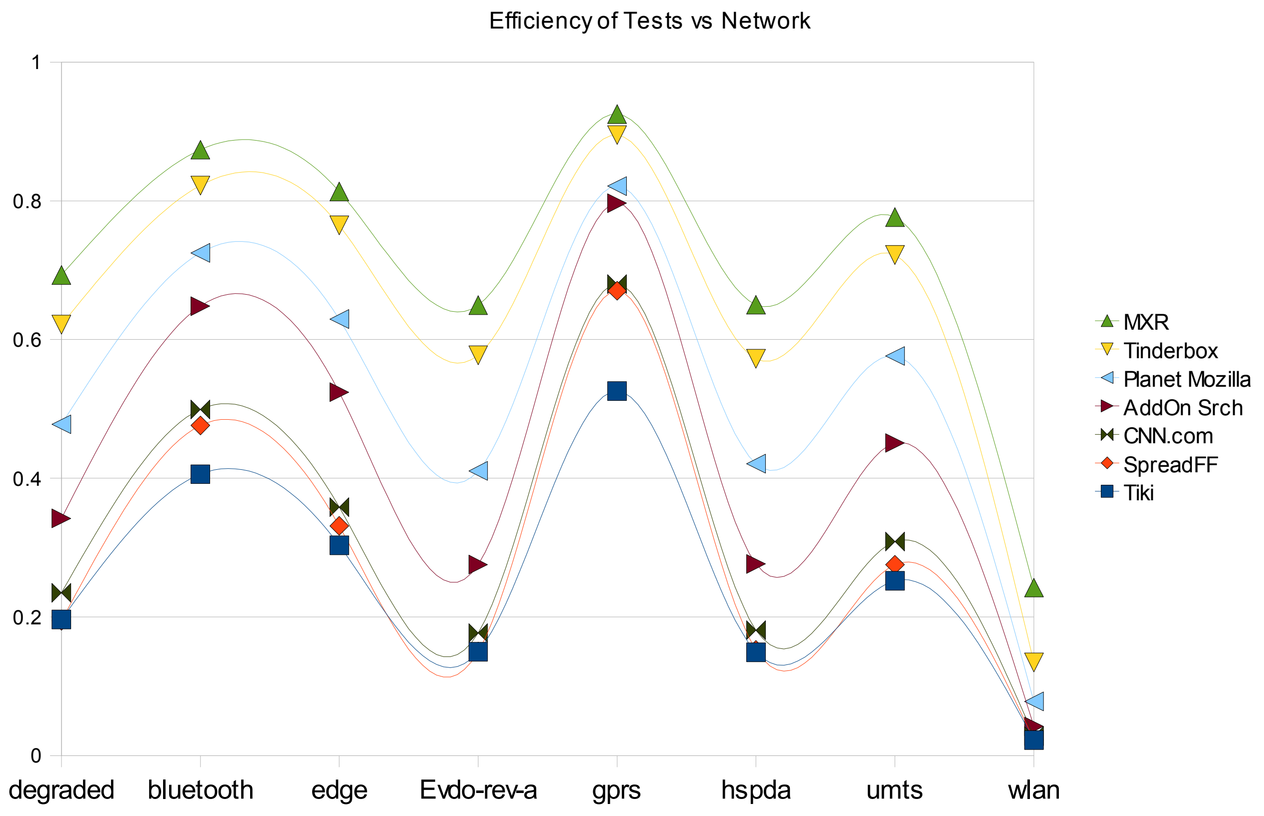
Focusing on improving efficiency metrics lets you focus on filling all available network capacity while not worrying about things you cannot control such as the provisioning of the network. It also normalizes performances comparisons across different types of networks.

The metrics are presented on the next page. There is one line for each website, with 8 points for each network type it was measured on.

The first good thing we see is that the graph is well ordered – the lines remain basically uncrossed. The most efficient sites are always the most efficient regardless of what network they are running on – and likewise the most efficient networks are always the most efficient regardless of what site they are transporting.

This hardly means things are as efficient as they could be – efficiency is a relative measure and there is no inherent reason that one network or site should perform worse than another. But the well ordered nature of the graph does mean we aren't seeing odd second order effects between test and network type which in turn reassures us that most improvement strategies will not involve harsh trade offs between test cases. That is a very desirable property as second order effects would make implementing any changes very risky.

EFFICIENCY RATIO MEASUREMENTS



There is an accompanying spreadsheet which has all of the individual data points and configuration characteristics of both the sites and networks.

Because the graph is well ordered, it is easy to tell which are the better performing test web sites. The order of performance (descending order: MXR, tinderbox, Planet Mozilla, AddOn Srch, CNN.com. Spread Firefox, and Niki-Forum) c**orrelates almost exactly with a sorted list of average object size** within each site (only the 6th and 7th entries on the list are out of order). It does not necessarily correlate with the number of objects on a page, or the total size of the page and included objects. **It is clear, and consistent with expectations, that the dominant factor in efficiency is the management of overhead of each HTTP transaction – and that this impacts some network types more than others.** The tools for addressing this are well known – request ordering, pipelining, persistent connections, etc..

Oddly, conventional wisdom would expect that the dominant factor in this transaction overhead would be network latency but that is not wholly consistent with the data. We would expect higher latencies lead to lower efficiency ratings. But that is demonstrably untrue in a number of cases – for instance the hspda network has 20% better latency and twice the bandwidth than the umts profile, yet it shows consistently lower efficiency numbers. (Remember, that doesn't mean it runs slower – just less efficient).   This, along with the shockingly low wlan numbers may just indicate a different bottleneck in higher bandwidth environments – though bluetooth behaved well and it shares the same bandwidth configuration as hspda. One last nail in the “latency is everything” coffin, the gprs tests have the highest latency in the whole test (900ms on average) and yet have the highest efficiency ratios.

Brief overview of Network Configurations:

Each configuration includes 70ms RTT of typical Internet wide area fiber latency on top of the latency for the wireless link.

- GPRS is so-called 2.5G technology. It is probably the least capable mobile network in wide deployment. It has bandwidth in the model of just 50/25kbit/sec (down/up) and enormous latencies of 900 (+/- 200)ms.
- Edge is the GPRS successor, sometimes called 2.75G. In the United States this is the exclusive network for the first generation iPhone. Bandwidth is 160/54 kbit/sec and latency is 770 (+/- 100)ms.
- EVDO-rev-a, sometimes known as CDMA2000 is a largely US based alternative to the 3G standards. It is modeled as 600/128 kbit/sec with latencies of 470 (+/- 75)ms.
- UMTS is so-called 3G technology common in much of Europe and Asia. It is modeled as 290/54 kbit/sec with latencies of 520 (+/- 100) ms.
- The last cell phone based network is HSPDA – which is an upgrade to UMTS. These are the networks commonly being planned and provisioned now, but not currently in wide deployment. This technology has the potential for large bandwidths depending on provider configuration, but it is modeled here according to common configuration: 700/500 kbit/sec with latency of 395 (+/- 75)ms.
- Bluetooth and 802.11 based networks are also interesting to mobile deployments. Bluet ooh is modeled at 700/400 with latency of 110 (+/- 10)ms
- Obviously a wide range of 802.11 conditions exist. This model uses a typical hotel or conference room value representing an extended 802.11g network: 24mbit/sec (symmetrical) and just 75ms of latency
- The last network profile is called “degraded”. It represents any of the higher bandwidth networks (3g, bluetooth, evdo) having a bad day. Bad days are not so much defined by bandwidth, but by latency spikes and packet loss. This network is modeled at a reasonable 175/54 kbit/sec of bandwidth but with delays of 1200 (+/- 300). This network also drops 4 in 1000 packets – which is a rate at least 8 times greater than the other networks.

Key Characteristics of Measured Networks

- MXR – This is a search result from the MXR tool. It contains the fewest objects in the

data set, just 3, dominated by a large piece of HTML. It has an average size per object 3 times larger than any other test.
- Tinderbox – This is a capture of the tinderbox status of firefox. It it a large data set, over a megabyte in total, spread across 22 different objects.
- Planet Mozilla – A snapshot of www.planetmozilla.org – the mozilla blog aggregator. This contains the most bytes of any test – about 1.5 megabytes and the second most objects (78).
- Firefox Add-On Search Results – The construction of this page contains lots of little images and icons, along with a large block of text.
- CNN.com – This page is almost always cited when soliciting mobile torture tests. It contains about 1MB of data, spread of 126 different objects including 2 flash files.
- SpreadFirefox Home page – Firefox's evangelism page is a surprisingly poor performer with lots of small images – containing 59 different objects over just 343KB of total data.
- Tiki Support Forum page – contains 25 objects over just 183KB of data. Suffers from the "aborted double fetch" problem below.

Other notes on Test Selection and Environment

- These tests may or may not share overlap with the "web 100" data set, but they serve different purposes. Web 100 is meant to be representative of normal browsing behavior, while these cases are all known to be "mobile difficult" and serve as meaningful guides to optimization.
- ***A medium term goal*** is to get this integrated into Talos. I still need a talos expert to help with that.

Metrics Beyond Efficiency

Usability may be defined as much by page load latency as it is by the total download time. I don't currently have a useful way to measure this and the time at which a page is ready to read sort of follows the "I know it when I see it" rule, which is hard to benchmark ;) Some possibilities:

- Time to first pixel displayed
- Time to complete rendering of viewable area
- Time to complete rendering of text in viewable area

Building a metric that reflects "page reading latency" is a strategic work area of investigation.

Specific Network Behaviors:

- **Persistent Connections –** the client was excellent, only closing a connection after a failed request (e.g. 404). Very little room for improvement here.

- **Pipelining –** The news on pipelining is mixed
  - Good News – Pipelining seems to be consistently enabled for sub objects on the page
  - Good News - If a pipeline is broken prematurely, unfetched objects from deeper in the pipeline are consistently rescheduled

        - Minor Bad News – Re fetches seem to be placed at the end of the queue, so they lose all priority they may have previously had
    - Bad News – Referenced objects are not placed in the request pipeline until the entire base page is downloaded. For instance, the Planet Mozilla page contains 77 referenced objects, but not a single one of them is requested until 10 seconds into the site load when the base page has completely been transferred. I haven't determined if the same is true recursively (for instance, an image loaded from a css), but it seems likely that it is. To overcome high latency concerns, the pipeline can be filled as soon as the reference is downloaded without waiting for the entire page. With respect to page re validations the revalidation of sub-objects can be pipelined along with the validation of the base object. ***This is identified as a strategic work area of investigation***

- **TCP sender congestion window –** It is likely that the high delays exhibited by some of the networks interferes with effective use of all available bandwidth, especially on the relatively higher bandwidth. Some additional work is still needed to determine how effectively those windows are currently operating – but it is likely this is a key bottleneck resulting in the relative under performance of umts, evdo, and hspda. ***This is identified as a strategic work area for investigation.***

- **Request Ordering –** Some included objects are more important than others, especially with respect to "page ready to begin reading" latency. An obvious starting point is separation of embedded JavaScript and css from media such as images, flash, etc.. and to prioritize the css and JavaScript. There is an obvious complication when combined with pipelining, but use of parallel requests queues could resolve this limitation. Furthermore, prioritization for elements impacting the currently viewable area allows asynchronous loading of off-screen elements. ***This is identified as a strategic work area for investigation.***

- **Parallelism –** No TCP parallelism was observed at all in my tests with a default profile. Parallelism could help overcome sub-optimimum bandwidth utilization associated with ramping of TCP congestion windows, and give more control over request ordering issues by allowing fennec to both pipeline and have some influence over head-of-line blocking problems. Advanced techniques such as shared control blocks and manual manipulations of receive windows would minimize overhead. ***This is identified as a strategic work area for investigation.***

Aborted Double Fetch Problem

One of the sites measured, the tiki based forum, exhibits what appears to be a strange bug in Fennec that has a performance impact. In this case the client requests the main page as usual. After downloading that HTML it pipelines requests for several embedded objects. Included in the pipeline is an unexpected second request for the original html page. Part of the way through receiving the response for the html page the second time the client aborts the connection – which means it needs to take the overhead of establishing a new TCP session. All of the objects in this test are served with "Cache-Control: no-cache" response headers. If that header is not present, the bug is not observed.

This is identified as a work item.

ROADMAP

Strategic Items

1. **Streaming Pipeline**
2. **Congestion Windows for large delay networks**
3. **Request Ordering**
4. **Parallelism and network scheduling of parallel flows**

Tactical Items

1. **Benchmark for Page Readability Latency**
2. **Talos Integration**
3. **Discussed “bridge mode” of simulator to allow more accurate measurements across a broader range of clients**