Chapter 4 – Performance Testing

While the implementation works and can help in detecting the features of requesting UA’s, its inherent goal is to improve the performance of web pages across both existing and future devices and UA’s. This chapter presents how we conducted our performance tests and their results.

# Mobile Web Performance

The performance of web pages has been steadily increasing along with the constantly improving network infrastructure and computing power of modern desktop computers. With this improved capacity Web developers have been able to create richer experiences on the web through increased use of resources such as CSS, JS, images and video. Because of this the size of Web pages has increased alongside the demand for more powerful computers to render them. With the emergence of mobile computing, Web pages again have to account for being viewed on devices with limited computing power, high network latency and reduced bandwidth. Web pages designed with desktop computers in mind can be orders of magnitude slower on a mobile device, sometimes to such a degree that the Web site is rendered unusable. This is in many ways a strange problem: on one hand mobile devices are much more powerful today than regular computers were ten years ago, but cellular networks are much akin to the regular wired networks in terms of speed as they were back then.

The web technologist, author and speaker Nicholas C. Zakas suggests in his article “The Evolution of Web Development for Mobile Devices” that the problems we face today with Mobile Web performance are rooted in two main concerns: network latency and mobile device hardware limitations. He bases his suggestions on the work of Steve Souders, who we mentioned in chapter 1, focusing on his list of best practices when it comes to Web performance, especially rule 1 (make fewer HTTP requests) and 11 (avoid redirects).

## Network latency

In the context of Web performance we define latency as the delay experienced from sending a request over the network to receiving a response, i.e. the round-trip time of the request. Bandwidth is defined as the amount of data a connection can receive over a specified amount of time, e.g. 20 Mb/s. Bandwidth may be limited by latency.

On regular wired connections, latency over short distances is minimal as the packets sent over the network propagate through physical cables. Barring network congestion, the only source of latency is electrical resistance in the wire material, or the speed of light in the case of optical fiber cables. As the transmission distance increases, so does the latency, because the signals propagate at a finite speed, but interference causing transmission loss is kept to a minimum. Desktop computers normally use a wired connection. This means that they will experience minimal latency when sending requests to Web servers, exacerbating the difference between making a request on a desktop computer versus a mobile device connected to a cellular network.

Wireless connections have several sources of interference and general signal loss that may increase latency when making requests to a Web server. Requests made over a wireless network propagate through the air, completely unshielded from any kind of external interference. Radios, microwaves, walls or any other form of electromagnetic or physical barrier may adversely impact the effective bandwidth of a wireless connection, giving wireless networks a much higher potential latency than wired networks.

Cellular networks are especially vulnerable to high latency by nature of their topology. A request from a mobile device has to first go to the nearest cellular tower, and then to a server using the General Packet Radio Service (GPRS) belonging to the mobile service provider, which functions as a gateway to the Internet. This server can then make the actual request to appropriate location (DNS, HTTP or other), the response from which then has to propagate the same way back. Currently these servers are few and centrally located, instead of distributed, giving the users proximity to them a measurable impact on the latency of a request. Going by Souders’ list of principles, rule 1 seems to be quite poignant here, as reducing the amount of requests will directly reduce the latency from the original request until the user sees a fully rendered Web page.

## Mobile device limitations

Modern desktop computers have extremely powerful hardware that have no problems with rendering even the most advanced Web pages. Even though modern mobile devices are quite powerful, especially compared to desktop computers from ten years ago, they do have limited processing power and memory compared to modern desktop computers. In this regard developers have to pay attention to how their Web pages utilize the capabilities of the device they are being viewed on. A Web page that is easily rendered on a desktop browser might cause severe problems or crashes on a mobile browser due to hardware limitations. The two things that are especially important is CPU and memory usage. CPU usage, as well as network access through cellular, Wi-Fi and Bluetooth antennas also impact the battery life of a device, which is already short on modern mobile devices.

# Method

Measuring performance of web sites is a whole research field in its own right. There are many ways of doing it, each aimed at specific parts. Some might target the performance on the backend, while others target the frontend exclusively, looking at the execution time of JS and the size of files sent in the response. Others may not look at response- or execution times at all, but rather do analysis of the content of a web page as it loads to determine the web page’s performance as experienced by the user.

Because our plugin is situated primarily on the backend we will focus on that as the common case, but we must also consider the case where the system encounters an unknown UA and must do tests on the frontend. These two cases are quite different and measuring their performance must thus be approached differently.

Enonic CMS also has its own device detection system built into it. Since our system is meant to replace it we also need to look at it and our system comparatively, to establish the performance impact of using our plugin as a replacement. This needs to be done for both cases mentioned above. Even though our system detects more features than the built-in system, a severe performance hit might be grounds to argue against using our plugin.

## Measuring backend performance

## Measuring frontend performance

## Comparing the built-in system with our plugin

# Results