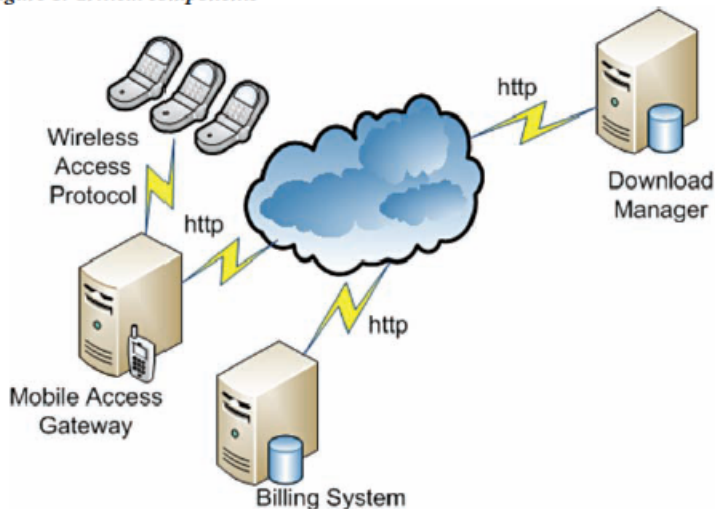


## Breaking Out: Load Testing With Java-Based JMeter

► The old saw “When all you have is a hammer, everything looks like a nail” bites even seasoned performance testers. It can take a little prodding and an out-of-the-box idea to remember we have other arrows in our quiver.

I was reminded of this recently when a performance tester friend—let’s call him Paul—asked if I could help on a project, load testing a mobile phone download manager. After expressing my immediate enthusiasm, I asked a series of pointed questions that revealed the system’s essential architecture (see Figure 1).

Figure 1: Critical components



A back-of-the-envelope analysis suggested that to apply load to the download manager, the obvious path was to use a tool that emulates wireless session protocols and develop scripts that simulate mobile phones browsing, buying and downloading ringtones, wallpapers and games.

My tool of choice for this project: HP LoadRunner and its wireless access protocol (WAP), with which I was both familiar and comfortable. Could I get access to the mobile access gateway (MAG) and have a go at it?

### The Plot Thickens

The answer, in a word, was no. The constraints: 1) Other test activities meant we couldn’t include an appropriately configured MAG in our test; 2) the number of mobile models we’d need to simulate was about 100, and there are significant differences in browser interface between models; and 3) the project budget meant licensing for LoadRunner, coupled with the consulting fees, would be out of reach. And, by the way, the customer’s initial testing had revealed performance issues that seemed to stem from the download manager, so the client was keen to eliminate other components and Internet latency from our test and focus strictly on the download manager.

Paul mentioned that the customer’s senior architect had done his

preliminary testing using Apache JMeter, simulating http requests aimed directly at the download manager. Hmm...we’d both been looking for an excuse to try JMeter ever since a friend at Google had shared her experiences, mostly positive, using it to test Google platforms. A quick discussion suggested it would be relatively easy to obtain a day or week’s worth of actual download manager traffic, as the system was already in limited production use. So that settled that—time to have a look at JMeter and a collection of weblogs!

### The Good...

The download service vendor was eager to isolate the location of the delays and immediately sent several days’ worth of weblogs to get us going. Paul crunched them through his data analysis Perl scripts and produced concurrency

graphs that revealed traffic peaking at about 45 concurrent sessions. We could see that request headers contained the handset model and billing number, and that about 20 handsets were represented. These logs provided the *actual* mobile user navigation paths and *actual* browse-to-buy ratios—no guesstimating or oversimplifying. With creative graphing, we derived realistic request pacing.

Delving into JMeter, I discovered that, while it’s definitely not as full-featured as LoadRunner, it would be able to handle this project’s requirements. Its terminology took some learning: *Thread Group* is a script scenario, *Samplers* are user request statements supporting various protocols, *CSV Data Set* is a parameter file and *Uniform Random Timer* is the construct for inserting randomized think-times.