



Improving End-User Performance by Eliminating HTTP Chattiness

Overview Microsoft® SharePoint®, Oracle® Portal, Microsoft Outlook Web Access, and Siebel® CRM 7.7 are just a few of the dynamic web applications critical to today’s organizations. But while users working near the corporate office’s data center have virtually instant access, remote or mobile users get painfully long delays, or even worse, find the application doesn’t work at all.

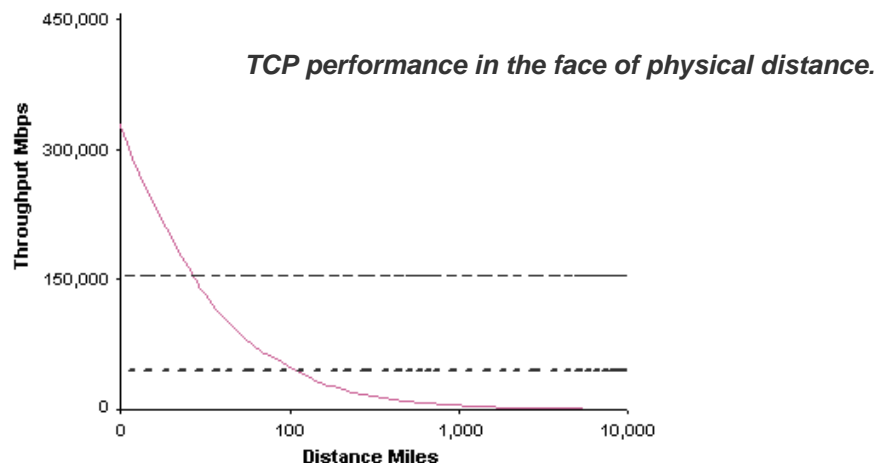
While application developers try rewriting the code, IT managers buy additional servers and bandwidth, all to no avail. New code and infrastructure can’t overcome the inherent problem of a chatty protocol traveling over high latency connections.

Challenge Many organizations embark on global multi-million dollar web application deployments only to later discover that their users are unsatisfied with the performance compared to their old client server application. In theory, widely distributed remote and mobile users can access applications from anywhere, increasing convenience and productivity. In reality, portals, CRM, collaboration and other enterprise web applications are delivered over the very chatty HTTP protocol. Instead of finding a nirvana of easily-accessible functionality, users are frustrated with slow performance and administrators struggle with low adoption rates.

Compounding the situation is the trend toward server consolidation. With web application users working from remote offices or mobile locations, it’s not uncommon to have users and servers on different continents, if not the other side of the globe. Even though data packets traveling at the speed of light are fast, latency due to the great distance traveled has a very real effect on web application performance:

Physical distance: It’s simple physics - a packet starting at the data center will get to a web browser 20 yards down the hall faster than it will to a web browser on the other side of the world. Even in a hypothetical situation where a packet is traversing an ideal connection over an uninterrupted fiber optic cable to its destination, there will still be a 200 ms (1/5 of a second) delay before it reaches its destination halfway around the earth. And to transfer one object via HTTP, it can take anywhere from 10 to hundreds of packets.

Network interference: In the real world, most data packets flow over wires (not fiber optic cable) and encounter many switches, routers, and other network devices, each adding a few milliseconds to the journey. If a device is busy, the latency increases. If the connections are low-bandwidth and congested, more delay is added to the round trip. As organizations look to send content over mobile and satellite networks, the packet is delayed even more.





Latency really becomes a problem when you are sending large amounts of data, like in the case of web applications, which commonly use 50+ objects to create a single page. And because each object requires that anywhere from 10 to hundreds of packets be sent across the connection, end users find that their applications are slow when the aforementioned factors are combined with HTTP protocol chattiness—caused by the browser’s conditional requests—which unnecessarily verify whether static content has changed.

of Web Objects * Chatty HTTP Protocol * Latency = Slow Application

Typically organizations try to accomplish web acceleration by caching static objects in a reverse proxy cache, offloading SSL from busy web servers to a network device, and offloading compression from the web servers to a network device. These types of acceleration will improve the time to generate the HTTP response if the web servers are overloaded, but other capabilities are necessary to improve delivery time over the network.

Solution Web application delays caused by network latency cannot be fixed by increasing bandwidth or increasing server capacity. F5’s WebAccelerator solves this problem with its Intelligent Browser Referencing (IBR) technology, which is a group of capabilities that eliminates the need for the browser to download repetitive or duplicate data, and ensures the best use of bandwidth by controlling browser behavior. By reducing the extra conditional requests and excess data (re)transmitted between the browser and the web application, IBR mitigates the effects of WAN latency, networking errors, and packet loss. IBR also significantly reduces the amount of data downloaded without requiring java applets or making changes to the browser. This is important, because it means IBR is a truly transparent technology. By comparison, other solutions that use applets destroy an organization’s ability to troubleshoot a site.

How does it work?

F5’s WebAccelerator changes the entire acceleration paradigm by safely caching content at the browser. Subsequent browser requests get the majority of their content from their own cache without the risk of using outdated objects. Use of bandwidth across the wide-area network drops dramatically and latency is altogether eliminated, since the browser is not making the request to the server (which avoids the subsequent network roundtrip).

Life before Intelligent Browser Referencing

The main reason web applications run slowly is the inherent chattiness of the HTTP protocol. The best way to combat this is to cache as many HTTP objects as possible at the browser. This eliminates the HTTP protocol’s chattiness and makes the web application much less sensitive to network latency.

The most common way to cache objects at the browser is to simply set long expiration times in the cache control headers of the objects. This is the approach used by many web development consultants, application server platforms, and other technology companies. This paper will contrast that approach to F5’s Intelligent Browser Referencing.

Legacy Approach: Setting Long Expiration Times

Traditional acceleration products combat HTTP chattiness by setting long expiration times in the HTTP object headers. Once the browser downloads an object with a long expiration time, it continues to use the object from its cache until its lifetime has expired.

This is achieved by:

Identifying all relatively static objects in a web application. This step requires in-depth knowledge of the web application's structure and usage pattern. Navigational images like arrow buttons and decorative items like company logos would be relatively static. However, recent news images would likely not be static.

Determining the proper expiration time for each static object. If the web application is updated infrequently, it might be appropriate to set lifetimes of one month or more. If the web application is updated more often, it might be appropriate to set lifetimes of one week. In many cases, even one day might be too long. Consider a web application with content that has a hard expiration date, such as financial information or auction information on an e-commerce site. In these cases, there could be adverse consequences if browsers use information that is even one minute out of date. Often, in the legacy approach, the decision as to the best expiration time must be arrived at over several iterations using trial and error. In addition, one might have to create many classes of objects with different expiration dates.

Drawbacks to this method include:

Complex implementation: It is difficult to maintain and is error prone.

Stale content: Once an object's expiration date is set, there is no way to change it.

Lack of control for third-party applications: Some packaged applications can be directly modified to set long expires manually, but still suffer from performance issues.

F5's Approach: Intelligent Browser Referencing

F5's WebAccelerator also sets long object expiration dates, but does so intelligently. WebAccelerator goes a step further by rewriting the object's URL to contain a checksum of the object's contents. At the same time, it maintains or establishes a short lifetime setting for the HTML page that references the objects.

When the browser wants to see if there is updated content, it re-requests the main HTML page from WebAccelerator since that page has a short expiration time. If nothing has changed, the browser receives the same main HTML page as before and then loads all objects from its cache since they were served with a long expiration time. However, if an object has changed, it is assigned a new object URL, which is reflected in the main HTML page. The browser receives a new HTML page with a different URL only for the object that has changed. The browser requests only that single object and then loads the remaining objects from its cache since they were served with a long expiration time.

This process has two very important advantages over setting long expiration times alone:

1. **Content safety:** Because the URLs have been altered to contain a checksum of the object, there is now a means of retroactively expiring updated content. HTTP objects can be safely given lifetimes as long as six months. If the content is updated, the browser will detect a new URL immediately and begin using the new information. This means the browser is never at risk for using stale content.
2. **Ease of configuration:** With no risk of using stale content, there is no need to worry about categorizing the HTTP objects of the web application into static and dynamic data. This allows the use of a very simple configuration such as simply looking for all mime-types that match images. There is no longer a need to have any knowledge of the structure or usage of the web application, or to consider how long to set the lifetime for the different classes of objects. A broadly applied six-month lifetime setting can be used. Even if the content changes every minute, the new content will instantly be available for all users to download as soon as the change is made.



Conclusion Today, organizations are experiencing two competing trends:

- Business-critical applications with highly dynamic content are becoming web-based
- Users increasingly access these applications from remote and mobile sources

These trends could work in tandem to increase productivity and profitability, but latency has crippled the applications and the business transactions that depend on them. Using the web application and content download is often so slow that the applications are useless in remote offices and for mobile workers.

Since latency can't be solved with additional bandwidth or server capacity, optimization and browser-based intelligence are the only options. Traditional tactics have relied exclusively on setting long object expiration times, which are complex to maintain and risky to dynamic content.

F5's WebAccelerator uses innovative Intelligent Browser Referencing to direct the browser to download only truly dynamic and unique data. Latency is mitigated by eliminating browser conditional requests for static data, while avoiding the risk to fresh and dynamic content seen in other solutions. The use of Intelligent Browser Referencing results in a low cost and transparent solution:

- Faster web applications
- Higher user adoption and satisfaction with business critical applications
- Simplified and faster application development
- Eliminates the need for additional bandwidth
- No need for extensive site management when content changes

About F5 F5 Networks is the global leader in Application Delivery Networking. F5 provides solutions that make applications secure, fast and available for everyone, helping organizations get the most out of their investment. By adding intelligence and manageability into the network to offload applications, F5 optimizes applications and allows them to work faster and consume fewer resources. F5's extensible architecture intelligently integrates application optimization, protects the application and the network, and delivers application reliability—all on one universal platform. Over 10,000 organizations and service providers worldwide trust F5 to keep their applications running. The company is headquartered in Seattle, Washington with offices worldwide. For more information, go to www.f5.com.