

ORACLE | PeopleSoft.

PeopleSoft® | Enterprise

# Oracle Enterprise 8.8 Application WAN Performance White Paper

December 2004 v1.0b

Stan Christensen

Performance & Benchmarking

Oracle, Inc.

## TABLE OF CONTENTS

1	Overview .....	3
1.1	WAN Benchmark Selection.....	3
1.2	Measuring Latency.....	4
1.3	Measuring Bandwidth.....	5
2	Backend Systems .....	6
2.2	System Utilization.....	7
3	Benchmark Transactions.....	8
4	Latency Results .....	10
4.2	Latency Result Summary .....	13
5	Bandwidth Data.....	14
5.1	Single User Transaction Bytes and Frames .....	16
5.2	Single User Transaction Bandwidth Utilization .....	17
5.3	Multiple User Bandwidth Utilization and WAN Saturation.....	19
5.4	Breakdown of Login/Logout per Transaction.....	21
5.5	Overhead when Compression is Disabled .....	22
5.6	HTTP Object Caching and Sharing .....	23
6	Summary .....	24
7	Addendum .....	25
7.1	Postscript.....	25
7.2	Transaction Detail.....	26
7.3	Transaction Object Detail .....	26
7.4	HTTP Cached Object Lifespan and Sharing.....	26
7.5	Single User Test Bandwidth Utilization Calculation.....	27
7.6	Load Runner E1 Bandwidth Utilization Data.....	27
7.7	CPU Utilization Detail Graphs .....	27

Oracle, PeopleSoft, PeopleTools, PS/nVision, PeopleCode, PeopleBooks, *PeopleTalk*, and Vantive are registered trademarks, and PeopleSoft Internet Architecture, Intelligent Context Manager, and The Real-Time Enterprise are trademarks of PeopleSoft, Inc. All other company and product names may be trademarks of their respective owners. The information contained herein is subject to change without notice. Copyright © 2004-2006 Oracle, Inc. All rights reserved.

# 1 Overview

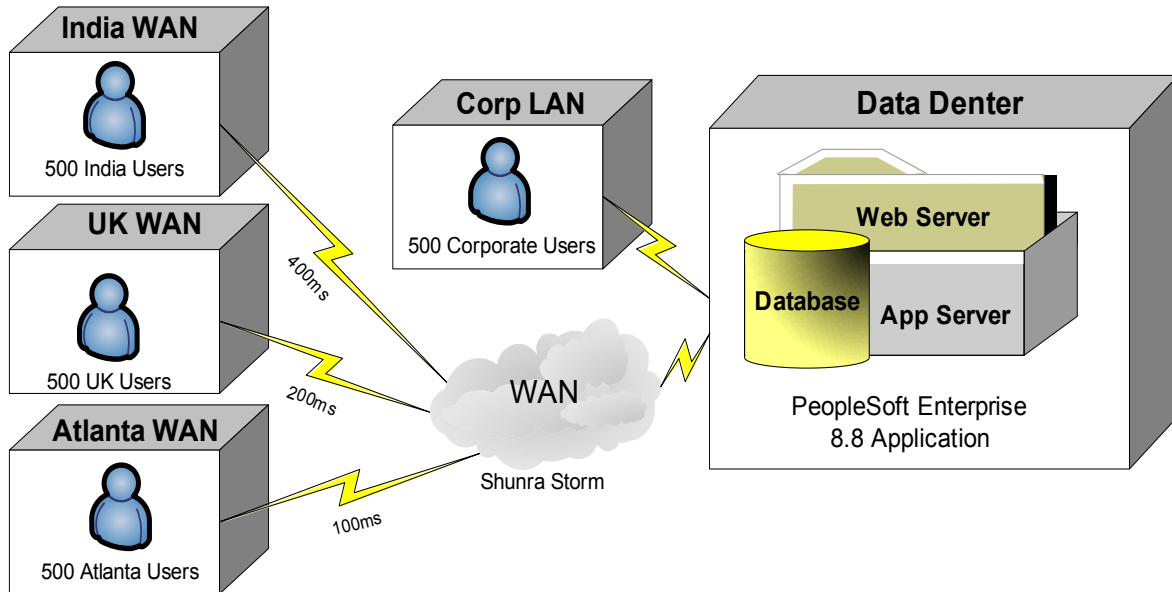
## 1.1 WAN Benchmark Selection

This paper is designed to provide summary of a large range of information on the details of Oracle (previously PeopleSoft) Enterprise and its network and WAN needs. Not all sections may be required to answer your particular interests, but it's important to note that the results detailed in one section may rely heavily on the groundwork laid in another section, such as the enabling of compression of the test traffic or the assumption of some browser caching. As this paper tries to be complete yet brief, there will likely be later white papers that expand certain sections in greater detail.

In providing a summary paper about the network performance and WAN requirements of Oracle's Enterprise 8.8 applications, we start with a complete and well-rounded real-world environment to do our testing. We've selected an implementation that would provide representative sets of transactions on a system running a typical load, utilizing servers and software that are appropriate for such an implementation. This was built on an existing 'Enterprise Portal 8.8 Benchmark' white paper and toolkit environment, with a mixture of HCM, CRM, and Portal transactions at a load of 2000 users. The environment benchmark and white paper is specified later in the addendum.

To add a global network, we divide the 2000 concurrent users into representative locales or sites that reflect several typical WAN characteristics. While replicating a global network can be done many different ways, depending on your goals and available resources, we chose the use of a US-based corporate headquarters LAN site for baseline comparison and the addition of 3 WAN sites representing the diverse locales of Atlanta, United Kingdom, and India.

### 1.1.1 WAN Benchmark Overview Diagram



And now we need to determine what would be useful network information about this real world environment we'd like to collect. For clarity sake (and to reduce the sheer possible permutations), we have limited our test to the universal constraints of latency and bandwidth only. This means we have not introduced such issues as out-of-order packets, congestion, caching, compression, and fragmentation. If these unique scenario issues are key to your network environment, it is recommended to measure them in your environment and use network predictive analysis tools to see how your particular issues would impact the traffic detailed later in this document.

Looking at our test factors, latency affects the 'responsiveness' of any network application, and knowledge of bandwidth is used to set and maintain adequate access between your clients and data center systems. Again, for clarity sake, we have treated these as separate issues so that we can focus on each in detail.

## 1.2 Measuring Latency

To determine the impact that latency has on 'responsiveness', we can directly use the transaction average response times reported from our 4-site global network environment (using a Shunra Storm WAN router/switch). The latency test scenario (using LoadRunner) is configured to generate 2000 simultaneous Web users performing a variety of transactions divided into our 4 locations. Our created network world has connections of <10ms, 100ms, 200ms, and 400ms latencies (represented as Corporate, Atlanta, UK, and India). As we're testing bandwidth independently of the LoadRunner scenario, we've limited all four sites to the high watermark bandwidth of E1 (2Mbit/s) per site. Early testing revealed that this bandwidth setting maintained an average peak utilization of just under half the E1 link (1Mbit/s) throughout the scenario test. This

provides a realistic real-world link type, yet eliminates congestion as an issue during our run time.

To create transaction response time averages on a per site basis, we've taken the transactions and replicated their scripts 4 times, each slightly modified to correctly select their site's network and corresponding E1 link. This created a scenario with an equally divided 500 Web users at each site, all in a single test scenario. These site-based transaction response times allow easy comparison on how increases in latency impact our application's 'responsiveness' across a single concurrent global scenario and are summarized later in this document.

### **1.3 Measuring Bandwidth**

Bandwidth is set simply by two factors: how much data you need to move and how quickly you need to move it. How much data needs to be moved for each of our typical transactions is easily determined, as Oracle Enterprise 8.8 relies only on HTTP as its client protocol, and the HTTP conversations vary very little between server platforms and network configurations in the Oracle Enterprise environment. We've captured all the Internet Explorer HTTP requests and responses that were made throughout the various testing stages of the benchmark and cross-validated them with the same HTTP conversations required by the LoadRunner Virtual User Generator (vuGen) HTTP client. This allows us to include an accurate, and completely representative transcript of all needed HTTP traffic for this environment's transactions. We've also included the TCP details on the transactions for reference, which would be representative of most corporate LAN environments. Again, it is recommended to use predictive network analysis tools if you would like to validate exact TCP details for your particular WAN or LAN conditions.

Now that we have the traffic for our environment, it is important to note where there may be some variables specific to you, starting with HTTP compression and caching. These are both determined by your combination of client and server. In our testing, we have measured Web server HTTP traffic to be virtually the same for both IBM WebSphere and BEA WebLogic, so server platform will not affect your HTTP traffic sizing notably.

Oracle Enterprise 8.8 applications have also been optimized for compression and caching by default. Virtually all HTTP response traffic is compressed if your browser supports it. If you have disabled this in your browsers, your Enterprise servers, or have legacy browser products, this will affect your WAN performance and bandwidth considerably. It is highly recommended to always leave compression fully enabled. Cache-control settings for most non-HTML objects are tagged for lengthy cache expiration periods by default, nearly eliminating HTTP 304 object modified requests. This means no constant trips to the server to see if it has a current object. This 'behavior' may also be changed by altering your browser settings, and/or version. We have left our Internet Explorer cache settings to 'Automatic' which is both typical and the most efficient use of network with IE. In LoadRunner, the cache and object request settings were set to make the traffic nearly identical to that of the Internet Explorer configuration. We have validated

through testing that the HTTP requests between Mercury's Virtual User Generator and Microsoft's Internet Explorer are nearly identical after the first iteration excepting one extra HTTP 'GET' request made by vuGen on some transactions (this is related to an additional navigational header rendering).

This benchmark is self-service transaction based, so it is important to note that we have assumed that the Web user's IE browser has already performed the transaction at least once, and that Enterprise's default 'deferred processing' component model is used. These assumptions mean a browser's cache is used for the requests to HTTP GET images, style-sheets, and GUI script, and that there's no need to continually return to the Web server to validate fields due to deferred processing behavior.

When adding up the bandwidth needs for your own organization, you'll need to include the additional overhead particular to your environment. If your users have not visited the Enterprise 8.8 site yet, they will need to cache a few image and GUI javascript objects. The addendum includes a complete breakdown on the objects needed for these transactions, which are relatively typical. Also you'll have your own particular overhead issues such as authentication (or single-signon) Web cookies, replacement or addition of corporate specified graphics, or customer changes to the Enterprise application such as reducing or increasing default 'deferred processing' levels. Any of these changes can greatly impact your particular network requirements.

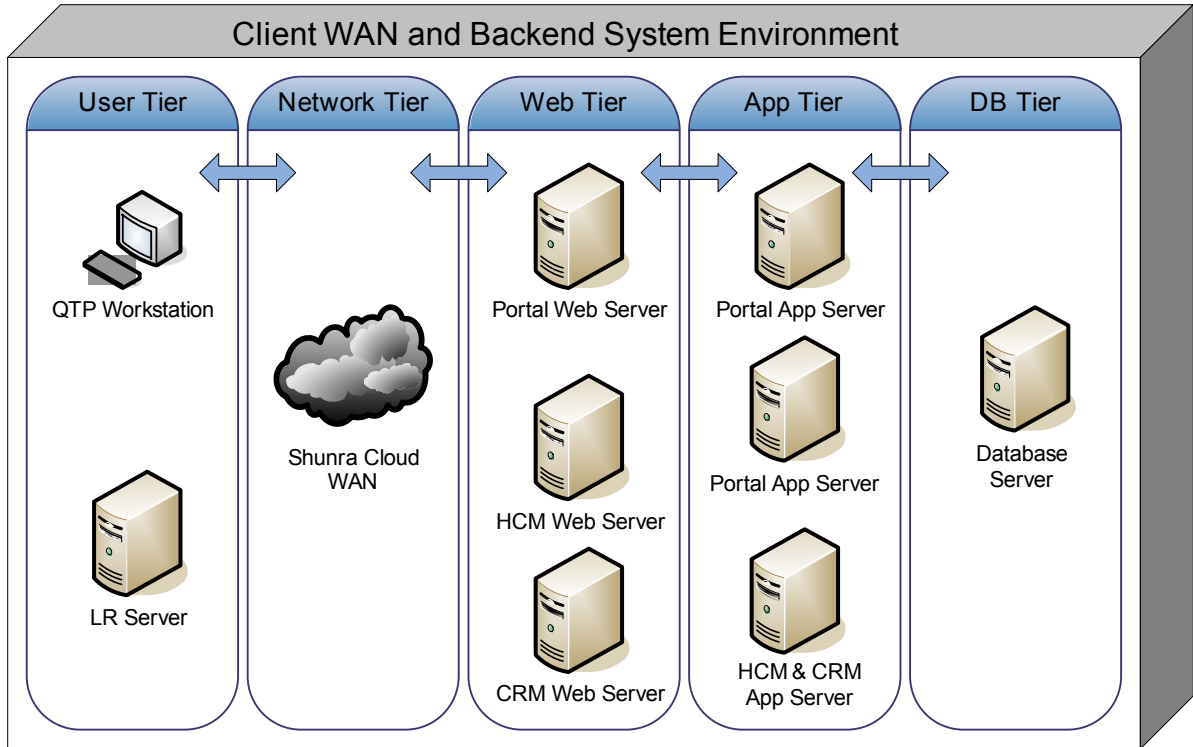
## 2 Backend Systems

As this is a WAN oriented testing, we focus almost completely on the relationship between your clients and the Web servers. But as this is a real-world environment generating a real-world load on our backend environment, we've included a brief description of the backend so that the transaction response times make sense. It is an obvious note that our WAN latency times would be much better if the system wasn't loaded and performing a typical set of transactions for our global environment. In this particular environment, we did not try to pick a load number that would have stressed our environment. We chose a load of 2000 users instead of a higher number to make certain the transaction response times we're typical, but not affected by limitations of the hardware or software (much like the choice of an E1 and its 2Mbps speed for our network link to prevent possible bandwidth limitations for the 500 users on each link). We've also included detailed CPU utilization of the systems in the addendum to get a sense of how much work this environment required of its servers.

The 'Enterprise Portal 8.8' environment used is a Linux-based environment with a single DB2 database server for all 3 database instances (HCM, CRM, and Portal), one application server for both HCM and CRM, two Portal application servers, and one Web server for each of the HCM, CRM, and Portal sites. Since this environment was designed for larger user capacity, and we have chosen to run it with a moderate 2000-user load, these systems do not approach their operational limits. Again, this is intended to keep the backend server environment as a 'non-issue' in detailing the network performance.

Complete implementation details are outlined in the benchmark’s toolkit and white paper mentioned later in the addendum.

### 2.1.1 Environment Diagram



## 2.2 System Utilization

The average CPU utilizations of the various servers are given below. The environment used local disk storage on the IBM xSeries systems and had memory sized for a benchmark that could run 4000 simultaneous users. Again, it should be noted that we purposefully ran the system with half of peak possible usage to eliminate the backend environment as a testing factor.

The CPU average utilization chart includes averages based on the one-hour stable period that is mid-scenario, and does not include ramp up and down times. This is the same period during which we ran our QuickTest Pro users for transaction times. Trended CPU graphs are included in the addendum.

### 2.2.1 CPU Utilization Chart

Average CPU Utilization	Number of CPUs	2,000 Users
Database Server	4	14.7%
Portal Application Server 1	8	12.7%
Portal Application Server 2	8	17.7%
HCM/CRM Application Server	2	8.3%
Portal Web Server	2	27.9%
CRM Web Server	2	1.0%
HCM Web Server	2	27.9%

## 3 Benchmark Transactions

This benchmark has a total of 7 self-service and kiosk transactions that are tested: 2 Portal, 3 HCM, and 2 CRM. All these transactions are accessed through the Enterprise Portal. A self-service transaction is defined as a user who does a *login* → 1 *transaction* → *logout* performed from a normal desktop. The kiosk transaction assumes the user will only be logging in from a shared kiosk system.

In the latency test scenario (using LoadRunner), it was setup to allow for a steady period of over an hour with each of the 2000 unique user's transaction being re-run every 10 minutes (Pacing). Think time within each user's transaction was limited to 5 seconds. The transaction definitions and scripts were left as they were originally defined in the source benchmark, which means that the placement of time measurement within each transaction (i.e. Login, Logout, or Search Results, etc.) between LoadRunner (LR) and QuickTest Pro (QTP) differ. This also means you may find an additional or omitted time measurement between the LR and QTP transactions. Normally a benchmark would not be presenting both numbers, which makes the differences moot. However, since a valid load is part of the latency test scenario, it is helpful to validate that LoadRunner is an effective tool in measuring WAN latency issues even though it utilizes an emulated Web browser instead of Internet Explorer. Since the WAN testing (for both LR and QTP) is using the LAN site as a baseline, our observations show that these differences between the browser types do not affect comparisons across sites. We discuss that further in later sections. However, it does limit comparisons for transaction times under LR and QTP to only their respective differences to their own LAN baselines.

It should also be noted that all of these transactions include login and logout, which, as you'll see in the bandwidth analysis, accounts for a large portion of total transaction bytes. For administrative or other transaction user types that remain logged in, simply ignore the login response times and back out the login/logout HTTP requests from the packet summary. We have included the transaction numbers broken down separately under the latency data, and provided the transactions with the size variations under the bandwidth section so that you may interpolate your own relevant transaction times for non-Self Service transactions.

While these transactions are all single-task focused, it should be noted that the additional overhead per HTTP page is typically only one or two GET/POST requests for each major component navigation (with deferred processing properly configured). This means that the need for network connectivity between your client and Web server is typically related directly to your navigation from major component (or page) to major component, not field to field, lessening the impact of latency on your user's experience.

Below are the seven transactions along with:

1. The number of users allocated to each of the four sites
2. The total number of users performing the transaction globally
3. The steps involved in the transaction

### **3.1.1 CRM Add Case (50 Users per site, 200 Globally)**

Login

Click on the 'Create New Case' link from the Help Desk pagelet  
Select Product details  
Click on Submit to save the case

Logout

### **3.1.2 CRM Select Case (50 Users per site, 200 Globally)**

Login

Click on CRM case ID number hyperlink from 'Self Service Help Desk' pagelet

Logout

### **3.1.3 HCM View Paycheck (125 Users per site, 500 Globally)**

Login

Click on view paycheck link to bring up users paycheck from shortcut pagelet

Logout

### **3.1.4 HCM View Benefits Enrollment (50 Users per site, 200 Globally)**

Login

Click on benefits enrollment link from shortcut pagelet

Logout

### **3.1.5 View Benefits Enrollment Kiosk (75 Users per site, 300 Globally)**

Login

Click on benefits enrollment link from shortcut pagelet

Logout

### **3.1.6 Portal User Profile (75 Users per site, 300 Globally)**

Login

Click on 'My System Profile' from the Enterprise Menu pagelet

Logout

### **3.1.7 Portal Content Search (75 Users per site, 300 Globally)**

Login

Enter the search word on the portal header

Search for the results

Select the respective link from the search results

Logout

## **4 Latency Results**

LoadRunner Analysis was used for reporting on all LoadRunner latency times, and log file time outputs from QuickTest Pro were used to measure Internet Explorer latency times. The 2000 user scenario's total duration is approximately 1 hour and 15 minutes and was run against BEA's WebLogic. IBM's WebSphere was also tested, but its data is not presented in the latency section, only the bandwidth section. All of the virtual users are activated within the first 10 minutes.

Once the steady state was reached with LoadRunner at 2000 concurrent users, the Internet Explorer automation scripts in QuickTest Pro were run on a dedicated Windows XP workstation. Each QTP transaction is run multiple times. The average response times are listed for all 4 locales for both LR and QTP separately.

### 4.1.1 LoadRunner Transaction Time Chart

Business Process	Latency	2000 User Scenario (LoadRunner Values in Seconds)*			
		500 Users (Corp) <10ms	500 Users (Atlanta) 100ms	500 Users (UK) 200ms	500 Users (India) 400ms
CRM Add New Case					
User Login		1.718	2.791	3.669	4.762
Add New Case		0.635	1.028	1.403	2.074
Select Product		0.183	0.444	0.655	0.979
Submit		0.867	1.008	1.433	1.245
User Logout		0.073	0.726	1.23	2.206
CRM Select Case					
User Login		1.779	2.791	3.71	4.795
Select Case		1.069	1.779	2.699	3.045
HCM View Paycheck					
User Login		1.514	2.289	3.306	4.515
View Paycheck		1.109	1.718	2.453	3.494
User Logout		0.073	0.625	1.008	1.807
HCM View Benefits Enrollment					
User Login		1.291	3.389	4.857	7.439
View Benefits		0.655	1.687	2.494	3.816
HCM View Ben. Enrollment (Kiosk)					
User Login		1.443	3.462	5.078	7.439
View Benefits		0.776	1.809	2.617	3.879
Portal Search Contents					
User Login		1.413	2.545	3.306	4.475
Search		1.21	1.646	1.921	2.067
Search Result		1.15	1.585	1.758	2.269
User Logout		0.043	0.575	0.968	1.767
Portal User Profile					
User Login		1.403	3.327	4.899	7.531
View Profile		0.726	2.054	3.183	4.259

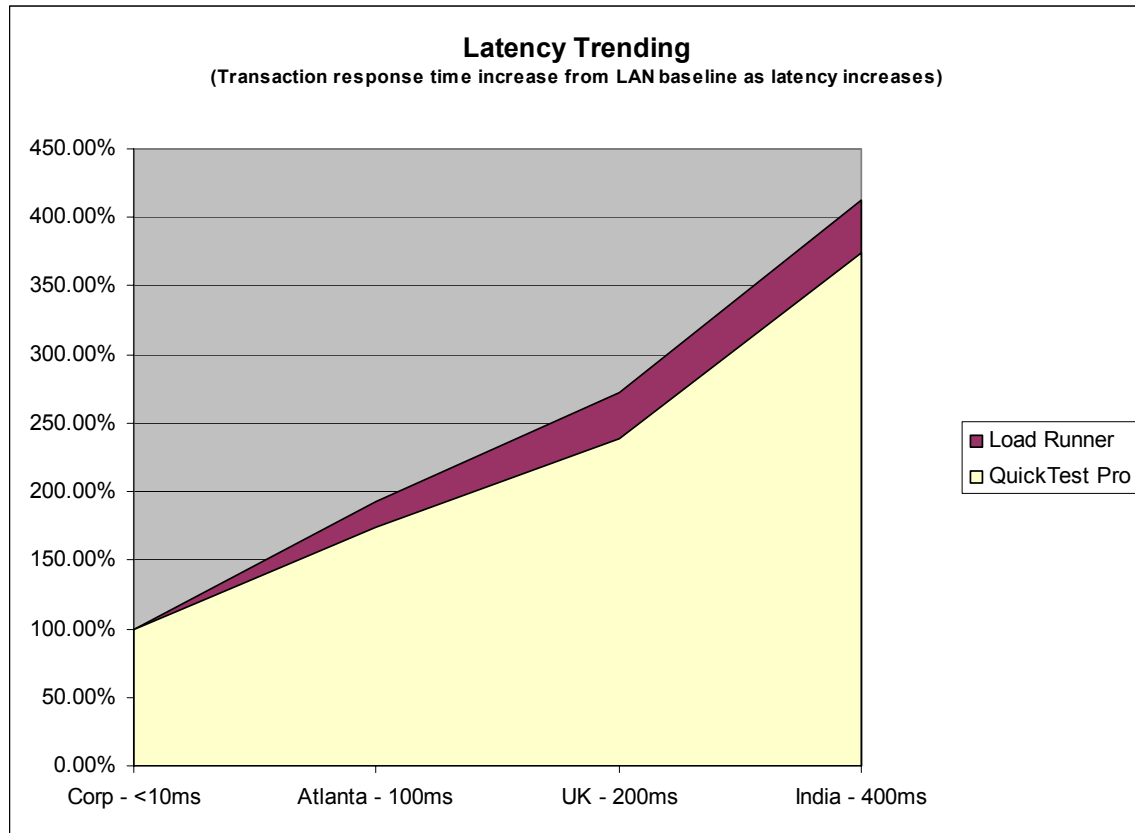
*\*Numbers adjusted for deviations not exceeding the 90% norm.*

### 4.1.2 QuickTest Pro Transaction Time Chart

Business Process  Latency	2000 User Scenario (QuickTest Pro Values in Seconds)*			
	500 Users (Corp) <10ms	500 Users (Atlanta) 100ms	500 Users (UK) 200ms	500 Users (India) 400ms
CRM Add New Case				
User Login	1.641	3.076	4.059	6.107
Submit	0.578	0.594	1.063	1.582
CRM Select Case				
User Login	1.681	3.102	3.870	5.798
Select Case	1.125	1.129	1.629	4.565
HCM View Paycheck				
User Login	1.547	2.669	3.803	6.085
View Paycheck	1.103	2.809	3.801	6.662
HCM View Benefits Enrollment				
User Login	1.181	2.780	3.689	5.270
View Benefits	0.909	1.596	2.216	2.720
HCM View Ben. Enrollment (Kiosk)				
User Login	1.466	2.492	3.482	5.643
View Benefits	0.794	1.505	1.967	3.226
Portal Search Contents				
User Login	1.372	2.759	3.834	5.065
Search	1.109	1.613	2.116	2.914
Search Result	0.053	0.056	0.063	0.063
Portal User Profile				
User Login	1.12	3.021	4.482	7.531
View Profile	0.569	1.658	2.553	4.259

*\*Numbers adjusted for deviations not exceeding the 90% norm.*

### 4.1.3 Latency Trending Graph



*(To make LoadRunner and QuickTest Pro times analogous, all Logout times were dropped from LR chart averages, as Logouts were not measured under QTP.)*

## 4.2 Latency Result Summary

In our tests, we can see an obvious and direct relationship between our transaction response times and latency. Using baseline response time as 100%, when our latency moves to roughly 100ms above our baseline (<10ms to 100ms), we have increased the transaction response time average to 183%\*. An increase in latency to approximately 200ms gives us an increased response time average of 256%\*. And an increase in our response time by nearly 400ms gives us an average response time of 393%\*.

What this shows is that while not quite a linear increase in average response, there is a reasonably direct and predictable increase for both our QuickTest Pro (Internet Explorer) and LoadRunner (Virtual User Generator, vuGen) times as our latency increases. Further testing has shown that this holds true as we approach 800ms (which was unable to be detailed in time for this test), but the trend line begins to turn more visibly upwards. It should also be noted that it might be typical for Internet Explorer to have a slightly longer transaction response time (due to its rendering overhead), but this is not the case in our testing. After verifying our results, there are two reasons for this.

The first is that our LoadRunner browser (vuGen) incurs an extra HTTP GET in some of the transactions due to its difference in caching/rendering methods when compared to Internet Explorer. Second, our QuickTest Pro box was not experiencing the same load as our vuGen box during our scenario. Regardless, what is important in our test is the comparison of the times between our LAN baseline and our WAN locales within each of our two testing tools against the same environment. The above results included both the Internet Explorer and vuGen response times to prove that the impact of latency is nearly identical for both.

It is also clear that Oracle Enterprise did not incur ‘chatty’ network conversations for any of these transactions. One typically considers an application ‘chatty’ when your client must, more often than not, make a request to the server with each click of entering data or navigating the interface. Here, our entire transaction conversations consumed no more than 5 to 12 HTTP requests (including login/logout, as seen in later sections) and therefore are affected less exponentially by the WAN than ‘chattier’ or non-deferred processing application transactions.

*(\* For the overall latency percentages of both LR and QTP, we’ve averaged both the Internet Explorer and vuGen times together.)*

## 5 Bandwidth Data

For our testing, we ran both Mercury’s Virtual User Generator (vuGen) Web client (automated through LoadRunner) and Microsoft’s Internet Explorer Web client (automated through QuickTest Pro). For bandwidth purposes, only Internet Explorer (IE) is directly relevant for exact bandwidth sizing, as vuGen is not a completely accurate browser with regards to its rendering and caching. However, since we are performing a comparison on both transaction times against a baseline, we will make a brief mention of how the LoadRunner vuGen traffic compares to our IE traffic in a few paragraphs.

First it should be mentioned which transaction conversations were summarized for our bandwidth numbers. From a network perspective, under Oracle Enterprise 8.8 applications, there are 3 notable browser conversation types:

1. The first type of browser conversation is on a completely uncached IE browser. This means that the IE client has never visited the Enterprise Application Web server at any time, and that the server will be communicating all objects needed to the destination IE client. This requires the most communication between the Web server and client.
2. The second type of browser conversation is a partially cached situation where the client’s IE browser has been used to navigate some portion of the Enterprise Application, but not for a particular user or application component that you will be navigating during your testing. This would be typical of an average user’s or Kiosk’s browser when trying to reach a different component or by a different user within the Enterprise Application.

3. The third type of browser conversation is on a fully cached browser for the transaction being performed. This means all images, stylesheets, GUI script, and navigational HTML are already on the IE client, so only a very small number of network HTTP requests are needed. This is typical where a system dedicated to a user is performing the same actions repetitively, such as a Procurement Officer who is constantly adding Purchase Orders into the Enterprise Application. This obviously generates the least amount of network traffic compared to our uncached and partially cached browsers.

The first browser conversation type (no cache) was not performed under our loaded scenario with Load Runner's vuGen, but was separately performed by our QTP scripts against an IE browser that had an empty file cache. The second browser conversation type (partially cached) is the default transaction run by our scenario under both vuGen and IE. The third browser conversation type (fully cached) is not detailed in either our latency or bandwidth sections, but can be interpolated by using the summarized data and the transaction detail in the addendum.

It should be noted that we tested little difference between the second (partially cached) and third (Fully cached) browser conversation types under both Internet Explorer and vuGen browsers when dealing with caches lacking user specific objects. As is mentioned later, a good portion of the needed objects is already cached if the browser has visited any component under the same Enterprise Application.

In testing, the difference between logging in with 2 different users performing the same task added exactly two additional HTTP GETs (per transaction under IE) when compared to the third type transaction (a fully cached browser). These two additional HTTP GETs under our partially cached browser (the LoadRunner/QTP scenario default) directly correspond to the retrieval of user specific navigational headers (at approximately 6500 bytes). Once these navigational headers were cached, the same user performing the same transaction was then fully cached (conversation type 3), and were no longer retrieved from the network.

When comparing the defined IE transactions HTTP requests (using our default partially cached browser conversation type) to the HTTP requests made by vuGen for our transactions under LR, we have validated that all are identical except the LoadRunner version of HCM Benefits Open Enrollment. This transaction performed one extra HTTP GET under vuGen than its IE and non-kiosk counterpart (due to its cache settings within LR). This means that the percentage differences between all our transaction times in the various locales can be considered comparable, and that our IE details are representative of what our LoadRunner scenario is generating.

The use of a partially cached browser (the second conversation type) as our default throughout the scenario is in line with our assumption that a typical Oracle Enterprise user has cached some portion of the related HTTP objects already, and most of our numbers below will reflect that. For instances where your defined transactions may have

fewer or more HTTP objects already cached, you can use the transaction detail in the addendum for reducing or adding HTTP objects and their corresponding network requirements.

It is important to note that only the latency testing section ran with transaction durations that resembled the real world (10 minutes). In this bandwidth section, all the transactions were run faster than humanly possible by using no pacing or think time under QTP automation of IE. A real user would not be able to type or navigate as fast as QTP, so the durations are for reference. Testing has verified that the HTTP bytes required did not change regardless of the duration taken for the transaction. The HTTP bytes moved during the 10 minute transactions in the latency testing match the HTTP bytes moved in the 30 second transactions recorded here. There is obviously some variation in TCP bytes due to the nature of keepalives, TCP's sliding window settings, and other TCP attributes, so it's important to note that any TCP numbers included here are typical overhead for a LAN environment.

## 5.1 Single User Transaction Bytes and Frames

We'll start with summarizing this scenario's 7 transaction TCP bytes and frames under the partially cached and uncached browser conversation types as directly measured under IE.

The first chart (5.1.1) is IE TCP details of transactions as they were performed under our scenario LR and QTP tests, with a new user performing against a partially cached browser. The second chart (5.1.2) is the IE details where the transactions are performed against a completely empty browser cache.

This latter chart (5.1.2) represents the maximum-case byte requirement on a greenfield browser, and is included here for reference. In most customer environments, an uncached browser would be an infrequent event such as a fresh system build for a user. It is important to note that in our testing, roughly a third to two-thirds of the HTTP objects needed are typically cached for any user's transaction, regardless of whether your particular transaction had been run, due to the amount of HTTP objects that are shared across Oracle Enterprise applications. These two charts represent the average to maximum byte requirements under our tested Enterprise transactions with IE browser cache set to automatic and compression enabled. The first chart byte sizes should be the most typical numbers for most all Enterprise customers, while the second is for reference.

### 5.1.1 Partially Cached Single User TCP Chart (Typical LAN Byte Sizes)

First user on partially cached browser	IE Duration (Secs)*	Bytes	Bytes from Client	Bytes from Server	Frames
CRM Add New Case	25.92	90479	23838	66641	177
CRM Select Case	22.68	63931	12077	51854	117
HCM View Paycheck	22.59	55013	11973	43040	104
HCM View Benefits Enrollment	21.69	52387	11893	40494	101
HCM View Ben. Enrollment (Kiosk)	22.60	52360	11917	40443	100
Portal Search Content	28.80	60558	14231	46327	115
Portal User Profile	22.09	51529	11470	40059	96

*\*As recorded by IE on non-loaded WebLogic environment with virtually no user think time added (back-to-back requests).*

### 5.1.2 Empty Browser Cache Single User TCP Chart (Maximum LAN Byte Sizes)

Empty Browser File Cache (Uncached Browser)	IE Duration (Secs)*	Bytes	Bytes from Client	Bytes from Server	Frames	% Increase Bytes vs. Partial Cache
CRM Add New Case	39.53	403649	98740	304909	665	<b>446.12%</b>
CRM Select Case	26.95	335514	76550	258964	534	<b>524.81%</b>
HCM View Paycheck	23.88	309432	79835	229597	500	<b>562.47%</b>
HCM View Benefits Enrollment	15.93	312695	83321	229374	508	<b>596.89%</b>
HCM View Ben. Enrollment (Kiosk)	32.60	313064	83385	229679	514	<b>597.91%</b>
Portal Search Content	23.28	244244	82804	161440	411	<b>403.32%</b>
Portal User Profile	17.59	241227	82242	158985	409	<b>468.14%</b>

*\*As recorded by IE on non-loaded WebLogic environment with virtually no user think time added (back-to-back requests).*

## 5.2 Single User Transaction Bandwidth Utilization

Bandwidth utilization is determined by the pace at which your client requests HTTP ‘GETs’ and ‘POSTs’ from the Web server and the number of bytes required to complete the request. This provides the two variables needed for bits per second (bps=(bytes\*8)/seconds). In our testing, we have all the details of how many bytes are needed for each of our transaction requests (chart 5.1.1), but our pacing will dramatically alter our total bandwidth needs.

Our bandwidth numbers were captured from the LAN with think time limited so that all HTTP requests are nearly back-to-back. This makes our overall TCP bits per second rates for our captured runs a ‘worst-case’ for the presented transactions, and these are presented in the last chart (5.2.3) with the exact numbers recorded.

In the real world, your users will be pausing between portions of these transactions to browse, enter data, or other normal user behavior and will be spending much longer than

the almost 30 seconds averaged in the testing runs recorded in chart 5.2.3. Our latency benchmark (in Section 4) defined a more realistic 10-minute (600 seconds) pacing for our transactions, so our first chart (5.2.1) presents the estimated TCP bits per second needed for our transaction with this same pacing.

Our second chart (5.2.2) presents the estimated TCP bits per second with 5 minute (300 seconds) pacing on the transactions (which is the most likely period over which one of these transactions would be performed), and therefore shows the increase in needed bps when the user performs the same transaction more quickly. This is still, obviously, a severe reduction in bandwidth requirements than for the 30-second runs.

### 5.2.1 **Partially Cached Single User Bandwidth Chart (with 10 minute pacing)**

TCP computed bits per second	Duration (Secs)	Total bps rate	Client bps rate	Server bps rate
CRM Add New Case	600	1206.4	317.8	888.5
CRM Select Case	600	852.4	161.0	691.4
HCM View Paycheck	600	733.5	159.6	573.9
HCM View Benefits Enrollment	600	698.5	158.6	539.9
HCM View Ben. Enrollment (Kiosk)	600	698.1	158.9	539.2
Portal Search Content	600	807.4	189.7	617.7
Portal User Profile	600	687.1	152.9	534.1

*\*Calculated with the Bytes as recorded by IE on a non-loaded WebLogic environment.*

### 5.2.2 **Partially Cached Single User Bandwidth Chart (with 5 minute pacing)**

TCP computed bits per second	Duration (Secs)	Total bps rate	Client bps rate	Server bps rate
CRM Add New Case	300	2412.8	635.7	1777.1
CRM Select Case	300	1704.8	322.1	1382.8
HCM View Paycheck	300	1467.0	319.3	1147.7
HCM View Benefits Enrollment	300	1397.0	317.1	1079.8
HCM View Ben. Enrollment (Kiosk)	300	1396.3	317.8	1078.5
Portal Search Content	300	1614.9	379.5	1235.4
Portal User Profile	300	1374.1	305.9	1068.2

*\*Calculated with the Bytes as recorded by IE on a non-loaded WebLogic environment.*

### 5.2.3 Partially Cached Single User Bandwidth Chart (with no set pacing)

TCP measured bits per second	Duration (Secs)	Total bps rate	Client bps rate	Server bps rate
CRM Add New Case	25.91612	27929.8	7358.5	20571.3
CRM Select Case	22.68287	22547.8	4259.4	18288.3
HCM View Paycheck	22.59239	19480.2	4239.7	15240.5
HCM View Benefits Enrollment	21.68567	19325.9	4387.4	14938.5
HCM View Ben. Enrollment (Kiosk)	22.60145	18533.3	4218.1	14315.2
Portal Search Content	28.79639	16823.8	3953.6	12870.2
Portal User Profile	22.09268	18659.2	4153.4	14505.8

*\*Calculated with the Bytes as recorded by IE on a non-loaded WebLogic environment.*

## 5.3 Multiple User Bandwidth Utilization and WAN Saturation

In our executed scenario that was used for latency testing (section 4), we set our pacing with each transaction performing once per 10 minutes. While this pacing was used to define the frequency that a transaction might be performed (once every ten minutes), it also refers to frequency that the transaction's bytes will traverse the network. Therefore 10 minute pacing means each transaction's bytes will be moved across the net over a 10-minute period and reflects an unintended network concurrency rate. With the scenario's 500 simultaneous users sending the transaction bytes every 10 minutes across its link, it is only about a 2-3 times lesser average bandwidth than an average user might complete the transactions (assuming 3-5 minute transaction lengths are more typical for our transaction set). The previous section provides a chart of what single user bandwidth would look like with the more typical 5-minute pacing (5.2.2). Here, however, we present the utilization as was actually recorded in our latency scenario.

As can be seen from the chart below (5.3.1), our 500 simultaneous users running at 10 minutes (600 seconds) per each executed transaction averaged about 686000 total bps.

That boils down to a rough average of 1.4 Kbps per user in our LoadRunner scenario. Obviously, individual users would be bursting above, and below this average, based on their particular pacing and think time.

### 5.3.1 Bandwidth Chart (Actual E1 link utilization for 500 users)

E1 link utilization with 600 second transaction pacing	Duration (Secs) per iteration	Concurrent users	Average bps rate	Peak bps rate
Corporate LAN Link (<10ms Latency)	600	500	669036	966591
Atlanta WAN Link (100ms Latency)	600	500	659898	1050473
UK WAN Link (200ms Latency)	600	500	670414	994951
India WAN Link (400ms Latency)	600	500	743243	1141265

*\*Calculated as recorded by WildPackets on our 2000-user loaded WebLogic environment.*

This leads very logically into the very natural question of how many users can I get for a given amount of bandwidth, or in different words, how many users does it take to saturate a WAN link.

While there were some PeopleSoft (now Oracle) white papers released on WAN usage some years ago, time and testing has shown that there is a much greater variance in the Enterprise application than was previously documented. This is partly due to an oversimplification of complex networks in the original white paper's testing, but more related to the evolution of the PeopleSoft toolset from PeopleTools 8.1 to its current versions. As with any product evolution, the current versions of Oracle Enterprise applications make much better use of network resources (using more accurate and better tuned caching, compression, deferred processing, etc.) and therefore have much higher rates of concurrency on network links than previous versions.

It is important to mention that your latency will impact your end user experience separately and in addition to the impact a saturated link might create. When confronting a saturated link, make certain you're considering any tools you might have to reduce both the bytes you're moving and its latency. You may find providing more bandwidth may not get the user experience to where you want due to the high latency typical of low bandwidth links.

If needing to predict the number of users you can fit on a link, such as a symmetric 56K ADN (Advanced Digital Network), you'll need a solid handle on both your user's pacing of the transactions they are working with, and the bytes each transaction needs to move. Also, any network link needs a degree of excess capacity for bursting and overhead, which a typical network capacity planner would set for 70-80% peak average utilization. That would make 42K available on a 56K link for any of your traffic, including the Enterprise application's traffic.

You'll also want to remember that most all WAN links allow duplex traffic (both network directions in use simultaneously), but these duplex links may be symmetric (same speed each direction) or asymmetric (different speeds each direction). For example, V.92 modem links allow download traffic speeds at up to 56K (though normally more like 52K), but they only allow upload speeds at up to 46.8K. The ADSL line in your small office or home is also likely asymmetric.

Our example, a 56K ADN, is 57344 bps each direction of capacity (and therefore a symmetric link), but is sometimes referred to by its total bandwidth capacity which in this case is 114688 bps. This measurement (Total bps) is also typically reported in bandwidth documents (and we have included them here), but it is each simplex bandwidth (the individual direction's bandwidth, such as your upstream versus downstream bandwidth) that will set your total symmetric rate. If the total needed bandwidth for your application is 100Kbps, but your client traffic requires 20Kbps and your server traffic requires 80Kbps, than you will need a symmetric link sized for the greater of the two values (80Kbps plus overhead in this example). To some degree, this makes the Total bps less relevant when sizing your network links.

For most Web applications, the HTTP traffic from the Web server to the Web client requires the greater bandwidth, and therefore is likely to determine what capacity a symmetric line will need to be sized with. This is true for Enterprise applications as well. You can see from the charts in section 5.2 that the client's bandwidth needs are predictably less than that of the server's bandwidth by roughly 20-40%. In the end, it does not matter whether your links are symmetric or asymmetric, only that you have taken into consideration each direction of bandwidth independently for both your application needs and WAN link sizing and are therefore not relying solely on Total bps results.

So predicting the saturation of our 56K WAN link from our 3 charts in section 5.2 would give us the range of an unlikely 50 simultaneous users from our latency test (Chart 5.2.1) to the impossibly bandwidth hungry needs of 2 simultaneous users in the QTP test (Chart 5.2.3). However, in the real world, you should see 4-40 active and concurrent users on such a 56K link, based on your user's caching, compression, duration spent on the transaction, and a variety of other factors mentioned throughout this white paper.

## **5.4 Breakdown of Login/Logout per Transaction**

As the above charts are the TCP values (as explicitly recorded from IE) representing self-service transactions (and therefore including login and logout), we will now breakdown the estimated overhead added to the bytes and frames for the login/logout process. These are computed by adding all the 'TCP bytes' and 'frames' related to Login/Logout under our default transaction, and then comparing them to the remaining core transaction bytes and frames.

You will notice that Login/Logout account for an overhead approaching 30-55% of the total bytes and frames. There are two primary reasons for this high ratio against our measured core transaction. Self-service transactions are typically single-step (and therefore single-component) oriented, so there is little more than 1-4 HTTP requests per core transaction request as compared to the very predictable 3 HTTP request for Login/Logout. Second, Enterprise uses its own compression engine that, on our tested PeopleTools version of 8.44, does not compresses packets before and after login. This means the number of bytes for Login/Logout is comparatively more, as can be seen in the complete transaction breakdown in the addendum. (It should be noted that there are a few exceptions to the HTTP request compression under Portal/CRM/HCM combined transactions, as specified in the addendum details.)

### 5.4.1 Login/Logout Overhead Chart

Login/Logout Overhead Only	Login/Logout Add. Bytes	% Additional Bytes	Login/Logout Add. Frames	% Additional Frames
CRM Add New Case	31310	34.60%	49	27.68%
CRM Select Case	31507	49.28%	53	45.30%
HCM View Paycheck	30782	55.95%	48	46.15%
HCM View Benefits Enrollment	30782	58.76%	48	47.52%
HCM View Ben. Enrollment (Kiosk)	30738	58.71%	47	47.00%
Portal Search Content	30791	50.85%	48	41.74%
Portal User Profile	30743	59.66%	48	50.00%

*\*Assumes our default partial caching and enabled compression under the IE & WebLogic environment.*

## 5.5 Overhead when Compression is Disabled

While it would be considered typical for any modern implementation of Web server and client to fully support HTTP response compression, we have included a further breakdown on the HTTP compression details for additional reference.

It should be noted that HTTP compression is only performed from server to client (as defined in the HTTP 1.1 standard), and is not performed client to server. Testing showed that our compression sizes did not vary between both of Enterprise's supported Web servers, BEA WebLogic and IBM WebSphere, as compression is performed directly by the Enterprise PeopleSoft Internet Architecture (PIA) code.

While one could test for the uncompressed packet sizes, the overhead for disabled compression is easily computed by taking the complete compressed TCP traffic sizes that were recorded and add the additional HTTP byte size increases of the uncompressed HTTP object (fully listed in the Transaction Object Detail addendum).

$$\text{Uncompressed TCP bytes} \approx \text{Compressed TCP bytes} + (\text{Uncompressed HTTP bytes} - \text{compressed HTTP bytes})$$

This means the additional TCP bytes and frames needed in overhead are not included, but still provide a reasonable estimate for any predictive analysis.

As can be seen, compression reduces total bytes considerably. While this is less of an impact on your LAN, it has considerable impact on your WAN where bandwidth and latency can be critical.

### 5.5.1 Disabled Compression Overhead Chart

Un-Compressed Bytes Overhead	Additional Uncompressed Bytes	Total Estimated Bytes	Estimated % Increase
CRM Add New Case	152283	242762	62.73%
CRM Select Case	78413	142344	55.09%
HCM View Paycheck	122871	177884	69.07%
HCM View Benefits Enrollment	84246	136633	61.66%
HCM View Ben. Enrollment (Kiosk)	84246	136606	61.67%
Portal Search Content	104847	165405	63.39%
Portal User Profile	79260	130789	60.60%

## 5.6 HTTP Object Caching and Sharing

As with any Web application, you want to share as many objects as possible across your various applications without degrading User Interface intelligibility. You also want a balanced use of object caching (with sufficiently long lifespan) so that an appropriate portion is cached up-front and remains current while the remainder of the objects are requested and cached as needed. This balance hopefully means you aren't waiting too long for objects you don't need and you're not waiting too long for functionally specific objects either.

Caching is determined by both your browser settings and the cache-control settings sent by the Web server with your object. Our Internet Explorer was set to 'Automatic', which means it will typically follow the Web-server's cache-control settings. The Enterprise applications typically sets the cache-control setting for non-dynamic objects with a 'MaxAge=315360000'. This means the object is always assumed to be current until it expires. You can see from the HTTP Cached Object Lifespan Chart (5.6.1) that this leaves few HTTP objects that will need to be validated or requested from the server if it has already cached the object. This is the preferred setting for the Enterprise application, as a different object name will typically be used when it's time for the object to be updated with another.

For object sharing, about a third of our objects are used by all our transactions across all applications. A slightly smaller portion of objects is used by each of our core application types (HCM, CRM, and Portal).

A complete HTTP object use detail, including relevant HTTP object settings is included in the Transaction Object Detail in the addendum.

### 5.6.1 HTTP Cached Object Lifespan Chart

HTTP object cache lifespan	Objects cached after any user	Objects cached for a particular user	Objects always requested
Total Number of Objects	114	12	22
Total Bytes	522369	71673	142084

### 5.6.2 HTTP Object Sharing Chart

HTTP object sharing	Cached objects used by all transactions	Cache objects shared by more than 50%	Objects shared by less than 50%
Total Number of Objects	45	9	94
Total Bytes	151797	32515	551814

## 6 Summary

PeopleTools is the foundation upon which all PeopleSoft's (now Oracle) Enterprise applications are written, fully Web-enabling our applications with similar Web caching, compression, and deferred transaction processing. This lends itself to equally similar WAN performance characteristics across our Enterprise Applications regardless of many backend changes, which can be clearly seen throughout our testing.

A major factor in this, and important to note when measuring your own performance, is that the Enterprise applications are defined and pulled from the database, and therefore all the HTML, image, and GUI script come from the application server. That eliminates most all content being statically defined on the Web Server, excepting a few key sections of static content.

While all of the environment definition that created this test environment can vary to some degree under your test constraints, we have taken great effort to choose all 'default' and 'typical' options to make this testing relevant to most customer environments. Never the less, the addendum contains the complete transaction and HTTP object details for a full view of all network and Internet Explorer requests that are made, should you need them.

It should be mentioned that a great deal of optimization has occurred with the evolution of PeopleTools to version 8.44, the tested version in this white paper, when compared to HTTP networking of earlier 8.X versions. Correspondingly, previous PeopleSoft networking white papers and benchmarks do not accurately represent numbers from the current versions of the Enterprise product. In boiling the optimizations down, our Web applications have been significantly improved to make sure we use as few HTTP requests as possible with the smallest required number of bytes possible. Both improvements in caching and deferred processing have greatly reduced the HTTP requests under the

Enterprise 8.8 applications. Compression and simplified object use have also reduced overall packet sizes.

This paper (and its addendum) provides complete and accurate HTTP byte sizing information of all needed traffic to perform these transactions. The only missing information to calculate your own environment's exact numbers is your network information and the pacing you'd set to the HTTP requests. With the recent advent of advanced network predictive software (such as Compuware's AppVantage), you should, in conjunction with a knowledgeable network team, be able to enter your specific network criteria and the specified transaction details to generate exact numbers for your environment. If you have extreme SLA commitments, this is obviously the wisest and most cost effective choice to guarantee the performance and sizing on your network.

Alternatively, if a real world test and data capturing is preferred, the test process that created this environment and benchmark can also be used as the structure for you to perform your own real world tests, as you might need. With the addition of minimal hardware such as Shunra's Storm WAN route/switch, you can reference the WAN Benchmark Testing White Paper detailed in the addendum for implementation details.

## **7 Addendum**

### **7.1 Postscript**

This is a white paper on WAN performance and network requirements of Oracle Enterprise 8.8 applications, and therefore contains only the information needed to present a coherent summary on the tested environment for brevity. For further detail, there are two additional papers that this summary is based on. None the less, many sections were kept purposefully brief for readability and later white papers will likely be released detailing certain sections with more background.

The implementation and environment is first based on PeopleSoft's (now Oracle) 'Enterprise Portal 8.8 Benchmark', which is still in draft. You can reference that benchmark and its toolkit for complete LAN implementation details when in release later this year. Secondly, there is the 'Enterprise WAN Benchmark Testing White Paper,' which is a 'how to' guide that outlines the additional equipment, including Shunra's Storm WAN Route/Switch, and the benchmark changes required to perform WAN testing. The Enterprise WAN Benchmark Testing White Paper is pending final draft release at this time.

The benchmark process uses Mercury's LoadRunner 7.8 and Quick Test Pro 6.5 to script and automate the defined transactions, but all references to browser behavior and captured HTTP and TCP packets are pulled directly from the network traffic and Internet Explorer 6.0 SP1 (for both Windows 2000 and Windows XP), not through any emulators, predictors, or Mercury Virtual User Generator.

In this benchmark we used Enterprise HCM 8.8 and Enterprise CRM 8.8 applications integrated with the Enterprise Portal 8.8 on PeopleTools 8.44, conducted on Red Hat Linux 2.1AS with an IBM DB2 database, IBM WebSphere Web server (for IE and bandwidth testing), BEA's WebLogics Web server (for LR and latency testing), and IBM® 'xSeries' server hardware. Packet capturing, packet analysis, HTTP capturing, and WAN switching were provided through the use of WildPacket's EtherPeek NX, CompuWare's Application Expert, SimTec's HTTPWatch, and Shunra's Storm respectively.

The remaining details of this addendum are primarily located in a separate Excel document. Each of the below section numbers correspond to an equally numbered tab in the spreadsheet. The text here serves to help explain a bit about the data you will find in the Excel document. Some sections such as CPU and Site Bandwidth have multiples of worksheet tabs representing their respective various servers or sites.

## **7.2 Transaction Detail**

The Transaction Detail spreadsheet (see the tab located in the Addendum Spreadsheet file for this white paper) details, per Transaction, all HTTP requests that occurred across the network with their TCP details. It does not include local HTTP requests served by the browser cache, as they do not generate network traffic.

The summary is captured by WildPacket's EtherPeek NX 2.0, and SimTec's HTTP Watch 3.2, while QuickTest Pro 6.5 is running the automated Internet Explorer 6.0 scripts. The TCP details are provided by 'post-processing' the network capture with CompuWare's AppVantage 9.5.

## **7.3 Transaction Object Detail**

The Transaction Object Detail spreadsheet (see the tab located in the Addendum Spreadsheet file for this white paper) details, per HTTP object, all HTTP requests that occurred regardless of whether they traversed the network. It therefore includes both the HTTP requests served by the local browser cache and the HTTP requests served by the network. The columns of each transaction will contain an 'X' if the HTTP object is used during the marked transaction.

The summary is captured by WildPacket's EtherPeek NX 2.0, and SimTec's HTTP Watch 3.2, while QuickTest Pro 6.5 is running the automated Internet Explorer scripts. Data verification is provided by 'post-processing' the network capture with CompuWare's AppVantage 9.5.

## **7.4 HTTP Cached Object Lifespan and Sharing**

The Object Usage and Caching Chart spreadsheet (see the tab located in the Addendum Spreadsheet file for this white paper) shows how the charts in section 5.5 were calculated.

It is based on the details provided for section 7.3, and therefore includes both the HTTP requests served by the local browser cache and the HTTP requests served by the network.

The summary is captured by WildPacket's EtherPeek NX 2.0, and SimTec's HTTP Watch 3.2, while QuickTest Pro 6.5 is running the automated Internet Explorer scripts. Data verification is provided by 'post-processing' the network capture with CompuWare's AppVantage 9.5.

## **7.5 Single User Test Bandwidth Utilization Calculation**

The SUT bps Calculation spreadsheet (see the tab in the Addendum Spreadsheet file for this white paper) shows the measured bytes per transactions, and the calculations for bps for both 10 minute pacing and 5 minute pacing. These numbers are referenced in section 5.2 of this paper. TCP byte overhead variations are not included and considered minimal.

## **7.6 Load Runner E1 Bandwidth Utilization Data**

The LR Site Bandwidth spreadsheet (see the tab specific to each site located in the Addendum Spreadsheet file for this white paper) shows the 30 second polled bandwidth utilization of our E1 links during the Load Runner 2000 user scenario. These numbers are referenced in both sections 4 and 5 of this paper. Note that only 500 users are running on each of the 4 sites.

## **7.7 CPU Utilization Detail Graphs**

The CPU Detail spreadsheet (see the tab specific to each server located in the Addendum Spreadsheet file for this white paper) shows the full CPU details for the Section 4 latency testing.