

SIQuA: Server-aware Image Quality Adaptation for Optimizing Server Latency and Capacity in Wireless Image Data Services*

Dong-Gi Lee and Sujit Dey
Department of Electrical and Computer Engineering
University of California, San Diego
La Jolla, California, USA
{synclee, dey}@ece.ucsd.edu

Abstract - Dynamic image adaptation techniques have been shown to be effective to provide required tradeoffs between image quality, network bandwidth availability, and wireless transmission latency, enabling low-cost, best-effort wireless image data transmission. However, dynamic image adaptation is a CPU-intensive task, and thus may cause servers to experience significant computational loads, leading to unacceptably high server response times, and poor end-to-end service latency. This paper investigates the problem of providing guaranteed server-side latency while performing dynamic image adaptation. We introduce a server-aware image quality adaptation (SIQuA) technique, used to optimize the runtime image adaptation process so as to provide guaranteed server-side latency with a maximized quality of image service level. SIQuA utilizes dynamic server conditions (server traffic loads and server resource availability), and optimizes both the quality of image serviced and the server-side latency required for runtime image adaptation simultaneously through judicious tuning of image compression parameters. We demonstrate that SIQuA can significantly reduce the server-side latency with a minimal impact on the image quality, thereby enabling efficient wireless image services with greatly reduced overall service costs.

I. INTRODUCTION

Wireless image data services, such as wireless Internet services with mobile phones and multi-media email services are rapidly gaining popularity and a larger share of the wireless data service market. However, enabling image data services over wireless networks requires more than a simple manifestation of the wireline data services on a mobile handheld device. For example, with the diversity in wireless access technologies as well as dynamic variation within each access technology, a user experiences a wide range of bandwidth fluctuations over time depending upon the user location and the coverage of user accessibility. Similarly, because mobile handheld devices vary significantly in their processing power, screen size, input mechanisms, battery capacity, and ability to present information (not all devices can support the image types or formats prevalent on the current Internet).

In order to address the above issues, several image adaptation techniques have been previously proposed [1-5]. For example, in our previous works [4, 5], we have presented dynamic image adaptation techniques that achieve runtime, optimal adaptation of image quality based on network bandwidth changes, appliance battery capacity, and user latency constraints. We have shown that our proposed dynamic image adaptation is very effective in enabling low-cost, best-effort wireless image data transmission. However, dynamic image adaptation is a CPU-intensive task, and thus may cause servers to experience significant computational loads, leading to a potential degradation of server throughputs, unacceptably high server response times, and therefore, poor end-to-end service latency. Especially, when high traffic load conditions exist at the server, *i.e.*, when the rate of incoming

service request is highly increased, the server cannot keep up with the requirements of the runtime image adaptation services, and therefore the server-side latency increases drastically.

In this paper, we focus on addressing server-side latency and server capacity issues incurred in dynamic image adaptation, using a novel *Server-aware Image Quality Adaptation (SIQuA)* technique. While most of the previously proposed image adaptation techniques target on addressing dynamic variability in network bandwidth or handheld resources, we found that dynamic service load variation at the server significantly affects end-to-end service latency. No other previous works that deal with server-side load variability issues during runtime image adaptation have been found.

The developed SIQuA technique exploits the main benefits observed in our previous work [4, 5] such that carefully tuning application-layer image compression parameters not only achieves an optimal tradeoff between image quality versus compression efficiency (therefore, addressing network-side latency issue), but also achieves optimal processing time savings during runtime image adaptation (thus addressing server-side latency issue). In addition to our previous work focused on addressing network-side variability issues (how efficiently customize image content in response to dynamically changing network conditions), our main objective in this paper is to minimize the impact of runtime image adaptation on the latency consumption at the server while maximizing the quality of image service level for each service request. Experimental results conducted using proposed exploration and evaluation framework indicate that the proposed SIQuA technique achieves significant server performance increases across a large range of service traffic loads and server resource conditions, with up to 74 % reductions in total server-side latency and 3X increase in server capacity while minimizing loss of image quality within 3 dB. Furthermore, the integration of the SIQuA technique combined with the network-centric adaptation technique demonstrates that adaptive image delivery service over wireless networks with end-to-end service latency guarantee is possible with addition of a low-cost SIQuA module to the server.

The rest of the paper is organized as follows: In the next section, we present background information about adaptive wireless image delivery services and the dynamic image adaptation techniques used by SIQuA. Section III describes SIQuA in detail. In section IV, we present experimental results conducted to evaluate the advantage of the SIQuA approach. Finally, section V concludes the paper.

II. BACKGROUND INFORMATION ON ADAPTIVE WIRELESS IMAGE DELIVERY SERVICE AND DYNAMIC IMAGE ADAPTATION

The overall service architecture and methodology for adaptive wireless image delivery services we envision is illustrated in Figure 1. For a given image request initiated from the mobile client to the image server, the main objective of this service methodology is to provide the most optimized (or customized) image content for the client through a runtime image adaptation process while adapting to the

*This work is supported by SRC under contract number 2001-HJ-900.

dynamic variability existing at the service environment. While most of image adaptation techniques focus on addressing dynamic variability that exists in network or client-side, as illustrated in Figure 1, besides network and client-side variability issues, dynamic server-side variation is other important factor for enabling customized image data services over wireless networks with low and predictable end-to-end service costs. For example, increasingly service request incomings (traffic loads) may cause the servers to experience significant computational loads, thus leading to unacceptably higher server response times and poor end-to-end service latency. Similarly, server capacity and server cost metrics also take significant portions in total end-to-end service capacity and costs.

We add a SIQuA module to the server so as to make an optimal image adaptation service in response to the dynamic service load variations at the server. SIQuA utilizes application-layer image tuning parameters and optimizes server-side latency through suitable tradeoff of the image compression results. In the next subsection, we describe how the wavelet-based dynamic image tuning method that is used by SIQuA can enable optimal adjustment of server-side latency.

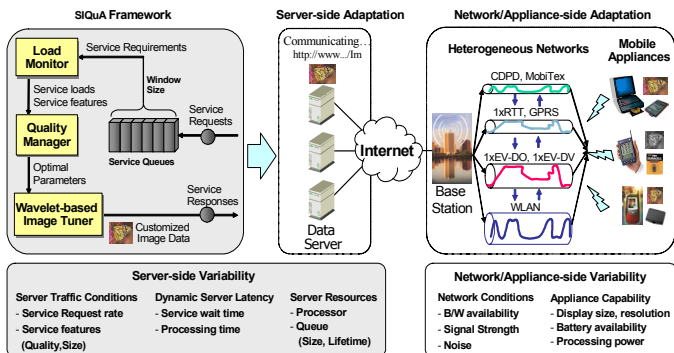


Figure 1. Overall service architecture for enabling customized image data services over wireless networks using SIQuA service framework.

A. Wavelet-based Dynamic Image Tuning

To make optimal image tuning at runtime in response to the dynamic variations in the service environment, the proposed wavelet-based dynamic image tuner utilizes three image compression parameters (transform, elimination [4, 5], and quantization levels) identified within the new JPEG2000 standard [6] as fine-grained application-layer image tuning parameters. In our previous work, the effects of varying each of image tuning parameters on processing time and energy requirements, compression ratio, and the image quality were quantified. Next, a proposed runtime reconfiguration algorithm [5] that utilizes the a-priori characterized information to decide the most appropriate image tuning parameters on-the-fly depending on the given service conditions and requirements. Note that the runtime reconfiguration algorithm takes into account the appropriate tradeoffs in the result of image compression and allows for fast reconfiguration (changing of image tuning parameters) with minimal overhead.

Figure 2 shows the effects of the proposed dynamic image adaptation on both processing time consumption (right y-axis, dark-line plot) and compression efficiency achieved (left y-axis, light-line plot). For each of the image quality we measured the minimal computation time required to satisfy a given image quality requirement (x-axis) using a 440 MHz SUN UltraSPARC 10 Workstation, normalized with the processing time spent to achieve maximal image quality (55 dB). Note that in addition to tradeoffs between the image quality versus size of compression efficiency, our proposed dynamic image tuning method also minimizes processing time consumption (maximum 75 % saving) depending on the image tuning parameters chosen.

While in this paper, we use this technique for addressing server-side latency and server capacity issues, more detailed information about

how we achieve this processing time savings during runtime image adaptation process and the algorithm itself can be found in [4, 5].

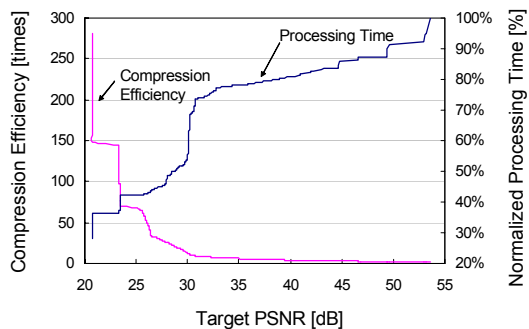


Figure 2. Effects of the proposed wavelet-based dynamic image adaptation technique on compression efficiency, processing time, and image quality.

III. SERVER-AWARE IMAGE QUALITY ADAPATION (SIQUA)

As described in the previous section, SIQuA is developed for addressing server-side variability issues in wireless image data services. It optimizes server-side latency through judiciously trading-off the selection of the image tuning parameters described in Section II.A. As illustrated in Figure 1, the key components of SIQuA framework are: (1) a load monitor, (2) a dynamic quality manager, and (3) a wavelet-based dynamic image tuner. The load monitor is responsible for gathering server conditions and service requirements such as traffic information (service loads and service features) and server resource availability (computing power and server queue size). This monitored information is then provided to the dynamic quality manager to determine optimal service parameters for runtime image adaptation so as to provide guaranteed server-side latency while maximizing the quality of image serviced for each service request. The wavelet-based dynamic image tuner is a fine-grained image compressor that executes runtime image compression to generate customized image content for the client request based on the optimal service parameters (image tuning parameters) selected by the dynamic quality manager.

The overall service flow using the proposed SIQuA framework is as follows: When an image delivery service request comes, its arrival time and information about its service requirements and network conditions are recorded. It is stored in a data server queue (we assume a FIFO service queue) and waits for its service turn for runtime image adaptation. The dynamic quality manager then retrieves servers' traffic loads and service features, and decides the optimal service parameters for each service request under the current server and network conditions observed by the load monitor. Finally, the wavelet-based image tuner customizes the requested image using the optimal service parameters chosen by the dynamic quality manager. The ultimate goal of SIQuA is to maximize the quality of the image serviced for each service request while managing server-side latency within a user acceptable latency bound. In this section, we introduce two server-aware image adaptation (or optimization) algorithms for controlling server-side latency: (i) an algorithm that optimizes quality of image service level on a per-request basis (*passive algorithm*), and (ii) an algorithm that optimizes the service parameters by looking at multiple service requests concurrently (*active algorithm*). In the following subsections, we describe both algorithms in detail.

A. Passive Algorithm

One possible approach to control server-side latency is to adapt the quality of image serviced on a per-service request basis. We first define the problem statement of passive server-aware image adaptation as follows: Suppose we have a set of service requests that arrives at a data server queue, $R = \{r_1, r_2, \dots, r_N\}$, where each service request contains its own image data service parameters such as service

latency requirement and the wireless datarate availability observed at the requesting client. Only one service request is processed at a time at the data server and the order in which they are processed is determined by a First Come First Served (FCFS) method. Each service request (r_i) is associated with a server arrival time (TA_i) and a server departure time (TD_i), and the time difference between these two time stamps represents its server-side latency ($TSlatency_i$). This time spent at the data server includes both the time spent for actual execution of the runtime image adaptation process (TP_i) and the service wait time in the data server queue (TW_i). Note that the time of departure (TD_i) is the sum of the processing time and the time of arrival ($TA_i + TP_i$) if there is no queue of service requests, or the sum of processing time and the time of departure of the previous request ($TD_{i-1} + TP_i$) when there is a queue of service requests that have already arrived, causing the service request to wait before processing. The objective of the passive algorithm is now to select the optimal image tuning parameters (TL, QL, EL), such that the server-side latency for each service request ($TSlatency_i$) satisfies given latency bound ($TSBound$), while maximizing the image quality serviced (measured in the Peak Signal-to-Noise Ratio, PSNR) for each service request. This can be formally stated as follows:

$$\begin{aligned}
 & \text{for each service request } r_i (i \geq 1), \\
 & \text{select optimal image tuning parameters } (TL, EL, QL) \text{ s.t.} \\
 & \text{maximize: PSNR}_i \\
 & \text{subject to: } TSlatency_i \leq TSBound \\
 & \text{and } PSNR_i \geq \min_PSNR \\
 & \text{where } TSlatency_i = TD_i - TA_i \text{ or } TP_i + TW_i \\
 & \text{and } TD_i = \max(TD_{i-1}, TA_i) + TP_i
 \end{aligned}$$

The algorithm first measures the time, the current service request had already waited in the service queue (TW_i) using the load monitor. The remaining time ($TSBound - TW_i$) can be used to optimize service parameters so as to achieve maximal image quality ($PSNR_i$) for each service requests. As observed in the previous Section II.A, since our runtime image adaptation can optimally tradeoff processing time required (TP_i), compression efficiency (CR_i), and image quality ($PSNR_i$) depending upon selection of the image tuning parameters, the goal of the passive algorithm is now to find the best parametric solution such that it achieves the best quality of image serviced (maximized $PSNR_i$) while satisfying the processing time (TP_i) and compression efficiency (CR_i) as service constraints through optimal parameter selection process as described next. Note that the PSNR of each image request is constrained by a “minimum image quality level (\min_PSNR)” as a lower bound in order to prevent over-regulation of quality of image service level. $TSBound$ is also set as the inter-arrival time monitored from the load monitor since we observed it achieves best-optimized server-side latency results under service load variations.

Runtime Optimal Parameter Selection: The quality manager in the SIQuA framework utilizes a lookup table called the TL-QL-EL table, as shown in Figure 3, for selecting the optimal image tuning parameters (TL, EL, QL) at runtime. Each entry in the table contains a-priori characterized image compression results, as observed in Section II.A, such as the image quality (PSNR), compression ratio (CR, represented as the original image size divided by the customized image size), and processing time per pixel (TP/P) for each combination of image tuning parameters (TL, QL, EL). We build this lookup table off-line using a large number of benchmark image samples and by extracting worst-case results that have the least compression efficiency (lowest compression ratio and the image quality for the given parameter combination) across a variety of the image samples. By utilizing worst-case image compression results, we can guarantee the minimum image quality level for all images requested. From the table, we see that for a given TL, as QL and/or EL increases, the PSNR and TP/P decrease, but CR increases. Also, for a given QL and EL, as TL increases, TP/P and CR increase. Note that

once the lookup table is built, it does not need to be changed during runtime parameter selection process. Only when there is significant quality mismatch observed during the service provision, it can be rebuilt. Therefore, the computational cost of generating the lookup table off-line is not a factor in the runtime parameter selection process.

To find the optimal image tuning parameters at runtime, we perform a binary search through QL (across rows of the table). We start the binary search at the first row (*i.e.* QL=0) which has maximal PSNR values. At each search step, we perform two searches along the row of the table (*i.e.*, along the QL values). One is with maximum EL values and the other is with minimum EL values for a given TL. Note that at each search step, all the TL values are checked to find the optimal solution. If a complete row search of EL=Max case does not satisfy the latency constraint, it means that the row of EL=Min case also cannot give any solution, and hence the QL value should be increased to the next binary search step ($QL = QL_{max}/2$). Similarly, when a solution exists for both EL=Min and EL=Max for a certain row (QL fixed), it means that QL can be further decreased for achieving better image quality (higher PSNR). The searching process is continued until a certain row with EL=Max has a solution, but EL=Min does not. In this case, we can find the solution by varying the EL values. Note that only the best solution, which means maximal PSNR (image quality) with satisfying latency constraints (TP/P and CR), would be selected (along with the corresponding TL, EL, QL parameter values). The worst case complexity of this algorithm is $O[(EL+TL)\log(QL)]$, which is not significant compared to the actual runtime image tuning processes (less than 5% of actual image tuning time taken).

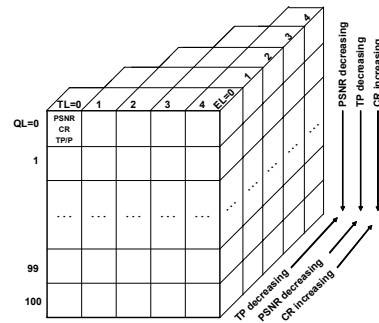


Figure 3. TL-QL-EL lookup table for optimal parameter selection.

B. Active Algorithm

While the passive algorithm attempts to meet the latency constraints for a single request, it does not consider the effects that the parameters it chooses have on the server-side latency of other service requests. This can lead to significant increases in server-side latency for some requests, especially if the request rate is increasing. Therefore, we next introduce an active algorithm that attempts to estimate what effect the parameters it chooses has, not only on the server-side latency for the current service request, but also for other service requests which are currently in the service queue at the server. A functional diagram of the active algorithm is shown in Figure 4. For each service request, the active algorithm first computes an initial solution using the passive algorithm presented above. It then estimates whether further reducing the server-side latency for the current request will be helpful for reducing large server-side latencies of other requests in the service queue. For further graceful adaptation, we also introduce an “efficiency factor (k)” that stands for the amount of efficiency the current adjustment of service quality level can achieve for the future service requests. It is defined as the ratio of the sum of server-side latency savings for other service requests waiting in the queue over the processing time reduction for the current request. If the efficiency estimated is greater than a given efficiency bound (Eff) then the passive algorithm is again applied to find the new sets of solution

(maximal image quality and its new set of image tuning parameters) for this updated server-side latency constraint. If the corresponding image quality found is higher than the given minimum image quality level (min_PSNR) set ahead, then new solution is updated. Otherwise, the previous solution is kept. This process is iterated until no further saving in server-side latency, or the image quality found reaches the minimum allowable image quality level (min_PSNR).

For the algorithm to work properly, given efficiency bound (Eff) must be set such that it is neither too sensitive (small value), nor too insensitive (larger value) for other service requests. If Eff is set too small, then the algorithm is very sensitive to a small change of the current processing time saving and thus has low efficiency. In contrast, if Eff is set too large, the required processing time reduction is too much to get the same amount of efficiency for future service requests, and therefore, may work as the passive algorithm. We have determined that a 10 % step size of reducing the current processing time, and an efficiency bound (Eff) of 3 have the good optimized results. While the complexity of the active algorithm is higher than the passive algorithm since it requires several repetitions of the passive algorithm, the active algorithm's overhead is still tolerable considering large latency taken for actual runtime image tuning process.

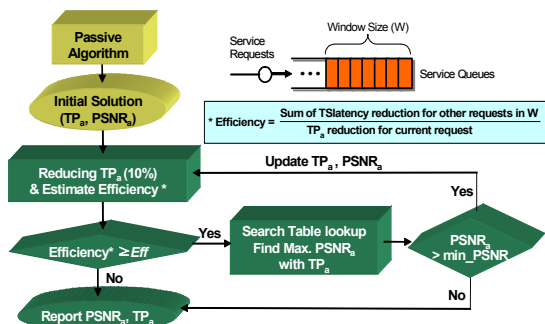


Figure 4. Process of active server-aware image quality adaptation algorithm.

IV. PERFORMANCE EVALUATION

In this section, we present experimental results that evaluate the effectiveness of the SIQuA approach under different service conditions. The main goal of these experiments is to support our claim that for wireless image data services better service can be provided by considering server-side variability issues. We also provide our experimental methodology used for evaluating different levels of image adaptation techniques on service performance issues.

A. Experimental Methodology

To evaluate the impact of SIQuA on server performance, we introduce a performance evaluation and exploration framework, shown in Figure 5. Our proposed framework is a simulation-based framework that evaluates server performance for different quantities and types of image requests. Since the SIQuA service targets next generation active image data services and there are currently no pre-recorded active image data service traces available, we generate analytical service workloads based on random distribution models.

To simulate the server performance, we first generate total 10,000 image service requests and their dedicated arrival times at the server using a Poisson distribution [7] (a common distribution used for representing communication network traffics). The properties of the Poisson distribution is defined by the average service request rate (number of incoming service requests per unit second). We also assigned a random wireless datarate and dedicated service features (or requirements) to each service request for the individual mobile user. These service workloads are used to evaluate the impact of SIQuA on the server performance while performing different levels of image adaptation tasks for each service request. We capture the performance

impacts in terms of server-side latency and capacity (or throughput), quality of image serviced, and server costs metrics.

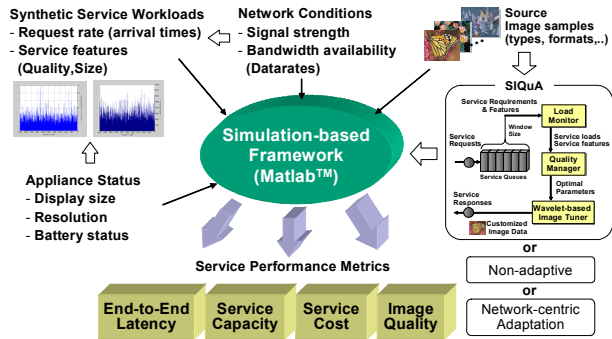


Figure 5. Proposed evaluation and exploration framework for investigating impact of different levels of image adaptation techniques.

B. Impact of SIQuA on Server Performance Increases

We first compare the efficiency of the passive and the active algorithms developed for the proposed server-aware image adaptation (SIQuA) technique. In Figure 6, we measure the server-side latency required (y-axis) to process runtime image adaptation for each service request (x-axis) under varying service load conditions at the server. A light-solid line represents the case of without server-aware image adaptation technique (i.e., only network-centric image adaptation applied) and a dark-dashed line represents the results of passive algorithm and a dark-solid line represents the results of the active algorithm. For convenience purposes, we zoom in on some specific requests. As noted in the middle of graphs (noted by Region 2), when the arrival rates of service requests are suddenly increased, the server-side latency drastically increases. We observe that this large server-side latency is mainly due to a large service wait time, waiting for its service turn in the service queue. Note that the passive algorithm can solve a lightly jammed server load situation (Region 1). However, if the server is heavily loaded, using the passive algorithm still allows large accumulated server-side latencies for increased the service request loads (Region 2). In contrast, the active algorithm monitors the service queue and detects a heavy load conditions ahead, and by preparing at the beginning of a heavy load situation appropriately, it can still maintain low server-side latencies even during heavy load conditions at the server.

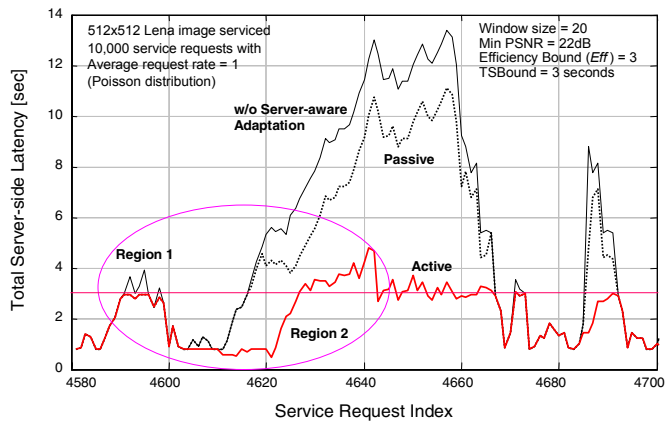


Figure 6. Impact of different levels of image adaptation technique on server-side latency measured for each service request.

We next evaluate the impact of SIQuA on the server capacity (or throughputs) and average loss of the image quality and report in Figure 7. We measure the server capacity as the number of service requests per second (service request rate) that a data server can handle

runtime image adaptation without dramatically increasing the server-side latency due to server overload. Under normal service-load condition, *i.e.*, when the rate of incoming service requests is below the capacity of the server's service handling, all incoming service requests can be processed without introducing large service delays. However, when high service load conditions exist (*i.e.*, when the incoming rate of service requests exceeds the server capacity of service handling), the runtime image adaptation process cannot keep up with the arrival rate of incoming service requests, and thus the service latency would be theoretically increased to infinity. While a client's tolerance for total service response delay is application dependent, typically a total delay of 1 ~ 10 seconds for web-based applications is considered acceptable [8]. If the total service response delay is longer than this threshold, the clients tend to lose interest. For our experiments, we assume that the maximum user endurable service latency is 10 seconds and 1 second is assigned for the server-side latency cutoff for extracting server overloading, the remained 9 seconds is reserved for network transmission to address last hop problem considering current slow wireless networks.

Figure 7 illustrates that without server-aware adaptation, it can only handle an average of 0.3 requests per second while the proposed server-aware adaptation technique can handle 0.51 (passive) and 0.89 (active) service request rate with an average loss of image quality of 0.8 dB and 3 dB, respectively. This experimental result demonstrates us that by adding small software module of SIQuA framework the server can provide significantly improved server throughputs with a minimal impact on the image quality loss.

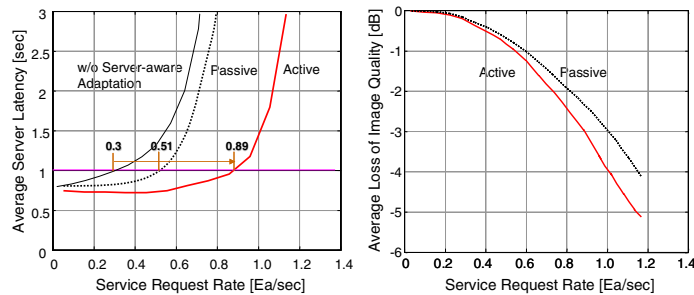


Figure 7. Impact of SIQuA on server capacity increases and average loss of image quality serviced for varying average service request rates.

Finally, we evaluate the efficiency of SIQuA from an end-to-end service point of view. We analyze the impact of SIQuA across different wireless access technologies to show the usability of our proposed technique under heterogeneous and multiple wireless network interfaces. Assuming 10 seconds is the soft end-to-end service latency (combining server-side and network-side latencies) constraint and under two different service load conditions (service request rates), we measure the percentage of end-to-end service latency violations (latency violated for total number of 10,000 service requests), average service latency, and aggregated quality of image serviced (PSNR) across different wireless access technologies (bandwidth availability). We deployed different wireless interfaces such as Sprint CDPD (19.2kbps) [9], Verizon 1xRTT (40kbps) [10], UMTS (384kbps) [11], and WLAN (1Mbps) [12] as reference access technologies. Each of this network system is currently available wireless access technologies, and we randomly generated dedicated datarate for each service request based on the average datarate marked as the above. Table 1 summarizes the experimental results.

As noted in Table 1, while non-adaptive (without any adaptation) and network-centric adaptation has significant portion of end-to-end service latency violation across different access technologies and incoming request rates, our proposed SIQuA combined with network-centric adaptation can drastically reduce the end-to-end service latency violation. This result demonstrates that our proposed SIQuA combined with network-centric adaptation can exist across multiple,

heterogeneous wireless network interfaces transparently, and thus can provide dynamic image adaptation services over heterogeneous wireless links with a guaranteed end-to-end service latency.

Access Technology	Average request rate	Non-adaptive Approach		Network only Adaptation		SIQuA + Network Adaptation			
		End-to-End service latency violation	End-to-End service latency (average)	End-to-End service latency violation	End-to-End service latency (average)	Average PSNR	End-to-End service latency violation	End-to-End service latency (average)	Average PSNR
CDPD (19.2kbps)	1 requests/sec	100 %	512.04 sec	97.62 %	12.45 sec	33.11 dB	3.26%	7.002 sec	27.37 dB
	2 requests/sec	100 %	512.04 sec	100 %	768.41 sec	33.11 dB	3.45 %	5.971 sec	25.21 dB
1xRTT (40kbps)	1 requests/sec	99.76 %	248.78 sec	92.89 %	12.06 sec	37.60 dB	1.58%	6.074 sec	29.72 dB
	2 requests/sec	100 %	248.78 sec	99.99 %	1159.0 sec	37.60 dB	1.68 %	4.745 sec	26.51 dB
UMTS (384kbps)	1 requests/sec	57.42 %	31.7 sec	28.59 %	7.15 sec	50.26 dB	0.08%	3.970 sec	38.19 dB
	2 requests/sec	100 %	31.7 sec	100 %	749.43 sec	50.26 dB	7.54 %	2.656 sec	31.28 dB
WLAN (1Mbps)	1 requests/sec	41.79 %	15.26 sec	11.91 %	4.59 sec	52.28 dB	0 %	2.512 sec	39.72 dB
	2 requests/sec	99.85 %	2013.4 sec	99.45 %	506.08 sec	52.28 dB	0 %	1.80 sec	32.68 dB

Table 1. Impact of SIQuA on end-to-end service results: Percentage of end-to-end service latency violation, aggregated image quality, and average service latency under different bandwidth availability and incoming request rates.

V. CONCLUSIONS

Future deployment of wireless multimedia data services will require very customized or personalized service methodologies, creating heavy workloads on wireless data servers. This paper presents a potential solution to the emerging problem by demonstrating the need of server-aware image quality adaptation to optimize server-side latency during active image data services. The proposed SIQuA technique optimizes server-side latency and quality of image simultaneously through appropriate tuning of application-level image compression parameters so as to provide significantly decreased servers-side latency with a minimal impact on the image quality serviced, thereby enabling content-rich active image delivery services over the wireless networks with a guaranteed end-to-end service latency.

REFERENCES

- [1] S. Chandra, C. S. Ellis, and A. Vahdat, "Multimedia Web Services for Mobile Client Using Quality Aware Transcoding," in *Proc. of 2nd ACM International Workshop on Wireless Mobile Multimedia*, pp.99-108, August 1999.
- [2] R. Han, P. Bhagwat, R. LaMaire, *et al.*, "Dynamic Adaptation in an Image Transcoding Proxy for Mobile Web Browsing," in *IEEE Transactions on Personal Communications*, vol.5, pp.8-17, December 1998.
- [3] A. Fox, S. D. Gribble, E. A. Brewer, *et al.*, "Adapting to network and client variability via on-demand dynamic distillation," in *Proc. of The Seventh International ACM Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS)*, pp.160-170, October 1996.
- [4] D. G. Lee and S. Dey, "Adaptive and Energy Efficient Wavelet Image Compression for Mobile Multimedia Data Services," in *Proc. of IEEE International Conference on Communications*, pp.2484-2490, April 2002.
- [5] D. G. Lee, D. Panigrahi, and S. Dey, "Network-Aware Image Data Shaping for Low-Latency and Energy-Efficient Data Services over the Palm Wireless Network," in *Proc. of World Wireless Congress (3G Wireless Conference)*, May 2003.
- [6] JPEG2000 - Our New Standard, <http://www.jpeg.org/jpeg2000.htm>.
- [7] S. Deng, "Empirical model of WWW document arrivals at access link," in *Proc. of IEEE International Conference on Communications*, pp.1797-1802, June 1996.
- [8] N. Bhatti, A. Bouch, and A. Kuchinsky, "Integrating user-perceived quality into web server design," in *Proc. of 9th International World Wide Web Conference*, pp.1-16, May 2000.
- [9] Sprint wireless Cellular Digital Packet Data (CDPD) services, <http://sprint.com/pcsbusiness/products/services/data/cdpd.html>.
- [10] Introduction of Verizon Wireless' Express Network (CDMA 1xRTT), <http://www.networkcomputing.com/1322/1322f37.html>.
- [11] Universal Mobile Telecommunications System (UMTS) White paper, <http://umts-forum.org/reports.html>.
- [12] IEEE, "Wireless LAN medium access control (MAC) and physical layer (PHY) specifications", *IEEE standard*, June 1999.