Elsevier Editorial System(tm) for Pervasive and Mobile Computing
Manuscript Draft

Manuscript Number: PMC-D-12-00106R1

Title: Quality of Information-aware Mobile Applications

Article Type: Fast Track IQ2S'12

Corresponding Author: Mr James Edwards,

Corresponding Author's Institution: The Pennsylvania State University

First Author: James Edwards

Order of Authors: James Edwards; Ahmed Bahjat; Yurong Jiang; Trevor Cook; Thomas La Porta

Quality of Information-aware Mobile Applications

James Edwards[a], Ahmed Bahjat, Yurong Jiang[b], Trevor Cook[c], Thomas F. La Porta[a]

[a] *The Pennsylvania State University*
[b] *University of Southern California*
[c] *US Army Research Laboratory*

Abstract

In this paper we propose a framework for providing Quality of Information (QoI) aware networking. QoI quantifies how useful information is for a given application. Its value is comprised of both *intrinsic* and *contextual* attributes related to the information. Intrinsic attributes include freshness and accuracy of the information. Contextual metrics include completeness and timeliness. To design QoI-aware network control algorithms, such as resource control algorithms, attributes of data must be mapped to QoI. Then, network algorithms may deliver the data in such a way that the ultimate information derived from the data is of sufficient quality for its purpose. We propose our QoI framework, and present the concept of QoI functions that capture tradeoffs between different attributes of QoI. We use optical character recognition (OCR) as a model image processing application. We focus on two attributes of QoI for the OCR application: accuracy and timeliness. We show how network controls and data processing, such as error recovery and compression, operating on data, impact the QoI delivered to a recipient. We then show how reductions in the required QoI may have a drastic impact on the amount of network resources required to support a QoI-aware transaction. Our results are based on an implementation on Android phones.

Keywords: Quality of Information, QoI, resource allocation, networking framework, tactical networks, timeliness, accuracy

1. Introduction

Resource allocation is critical for applications running on constrained networks. Unfortunately, data communication network resource allocation algo-

[☆] Research was sponsored by the Army Research Laboratory and was accomplished under Cooperative Agreement Number W911NF-09-2-0053. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Laboratory or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation here on.

rithms do not typically operate in an information-sensitive manner, but instead focus on the delivery characteristics of bits of data. We propose the notion of Quality of Information (QoI)-aware networking.

QoI is a composite, multi-dimensional measure that quantitatively and qualitatively identifies the degree to which data is suited for an application or a decision making process. Unlike Quality of Service (QoS), QoI considers both *intrinsic* and *contextual* aspects of information as discussed in Section 2. Intrinsic attributes include things like the age of information and its precision. Contextual attributes include things such as timeliness and completeness. QoI-aware networking algorithms must consider how data is transformed into information and the impact that the network may have on this transformation.

As a simple example, consider a photo as a piece of information. The *intrinsic* attributes include freshness (age), the time that the photograph was taken. The *contextual* attributes include timeliness, which may be considered if the photo is being transferred across a network. In the case of a photo, timeliness may be improved by cropping or compressing. While we may instinctively think of these operations as degrading accuracy, this is not necessarily the case. For example, a photo with large amounts of unnecessary data may be cropped without affecting accuracy and a high-definition video may be compressed if the content of interest is sufficiently large and recognizable. These tradeoffs are made explicit in QoI-aware networking through the use of QoI-functions.

In this paper, we address the quality measure trade-offs in constrained communication networks for image applications. We use optical character recognition (OCR) as an example image processing application. We discuss QoI functions for this application to illustrate the trade-offs of different QoI attributes. We lay the foundation for a general model of QoI-aware networking to optimize information flow based on a comprehensive understanding of the information and the communication channels in use. As part of this, we map information quality attributes to data quality attributes, and determine the impact of the network on the delivered QoI. This model can be applied to information media other than images, such as video and audio.

In addition to simulation results, we present experimental results from an Android mobile phone implementation. Because of the reliability of mobile wireless links, we consider Forward Error Correction in addition to various image transformations. Our results show that this QoI-aware approach allows more queries to be answered, faster, and with less network impact.

The rest of the paper is organized as follows: Section 2 details the QoI model. Section 3 describes our experimental setup and Section 4 presents results. Section 5 discusses related work and Section 6 concludes the paper.

2. QoI Model

The QoI model we propose is applicable to both information at rest (stored) and information being retrieved, either in real-time or from storage. QoI is tied to the data used to generate the information and includes both intrinsic and contextual attributes as we discuss in detail below.

We define both a *requested QoI* and a *delivered QoI*. The requested QoI may be specified by the requestor as a vector of attributes, or a function that specifies the relationship (tradeoffs) between the different attributes. The delivered QoI is what is received by the requestor, perhaps after transfer across a network or after transformations have been performed on the information. The delivered QoI is stochastic because often the impact of transformations of data (for example compression) on information are not perfectly predictable; likewise, network conditions may vary and cannot always be accurately predicted.

In this work we distinguish data from information. Information is derived from data. For example, if an image is used to determine the number of people in a room, the data is the image file and the information distilled from the data is the value of the number of people. Below, in general, we use the term information in our discussion. When data must be distinguished from information we do so explicitly.

2.1. QoI Functions

QoI functions specify the relationship (tradeoffs) between different information attributes. The attributes are directly related to the information as it is being used by an application. The attributes can be classified as being either *intrinsic* or *contextual*. Intrinsic attributes are inherent in the information regardless of its use, for example the freshness of information, the bitrate of an audio sample or the resolution of a photograph. Contextual attributes depend on its use, for example the completeness or timeliness of information.

In general we can express the QoI as determined by an application as:

$$QoI_{\text{APP}} := f(attr_{\text{INTRINSIC}}, attr_{\text{CONTEXTUAL}}) \tag{1}$$

where $attr_{\text{INTRINSIC}}$ is a vector of the intrinsic quality attributes and $attr_{\text{CONTEXTUAL}}$ is a vector of the context-specific attributes.

The data delivered to an application has a certain fitness for use and can generate information of a certain quality. We call this measure the application score, S. For example, given an image of a face, a face recognition algorithm can match the face to a known person with an expected certainty. This certainty is the application score of the face detection algorithm.

Similarly, throughout this work we use OCR as an example application – the application score in this case is the degree to which an OCR application's output matches the actual document's contents. In both cases, however, additional factors, both intrinsic and contextual, impact the final quality of the information. For example, the information may be required within a certain time frame, thus imposing a contextual timeliness requirement. If it is received late it may not be as useful or of any use at all. We can therefore rewrite (1) for our example as:

$$QoI := f(S, A) \tag{2}$$

where S is the application score and A is the impact of other contextual and intrinsic attributes.

To evaluate QoI, one must consider the quality of *data* used by the application. Then, a mapping between data quality and information must take place. For example consider an OCR application: The accuracy of the OCR application is a function of the image (data) received. The image will have a certain precision (bits per pixel) and accuracy (errors). Thus the accuracy attribute of the information is a function of the precision and accuracy attributes of the data.

Similarly, one can consider the information attribute of timeliness. This is a function of any preprocessing time, the size of the data being transferred and the rate of the channel. The size of the data being transfered is a function of the type of data and its properties (for example whether it has been compressed or if overhead has been added to correct transmission errors). Thus, some of the data attributes that impact the timeliness of the information (compression and error recovery methods) also impact the accuracy of the information.

Below we discuss several information attributes and the data attributes on which they depend. We then instantiate an example using an image application.

2.2. Information Attributes

In this section we discuss timeliness, accuracy and precision of information, and the data attributes on which they depend. We also briefly discuss some other information attributes.

2.2.1. Timeliness

Timeliness is a measure of meeting a deadline by which information is required. Timeliness functions are simply a way to express the timeliness measure explicitly, as a function of time or response delay. Examples of timeliness functions include step functions, linear decay with time, and functions with hard and soft deadlines ("double deadline"). The double deadline function can be expressed as:

$$T(t) := \begin{cases} 1 & \text{if } t \leq d_1 \\ \frac{1}{d_2 - d_1} \times (d_2 - t) & \text{if } d_2 > t > d_1 \\ 0 & \text{if } t \geq d_2 \end{cases} \tag{3}$$

where d_1 is a soft deadline, d_2 is a hard deadline and t is the expected end-to-end delay of the information. An example of a double deadline is shown in Figure 1.

The timeliness function is itself contextual. The specific value of timeliness is impacted by both network attributes and data attributes. The network rate and size of the data being transferred directly impact the timeliness of the information. In turn, the size of the data is impacted by the data source, transformations on the data such as compression, and any overhead added by network protocols. This overhead may cause increased data volume, for example if extra bits are added for error detection or recovery, or increased transmission time if retransmission protocols are used.

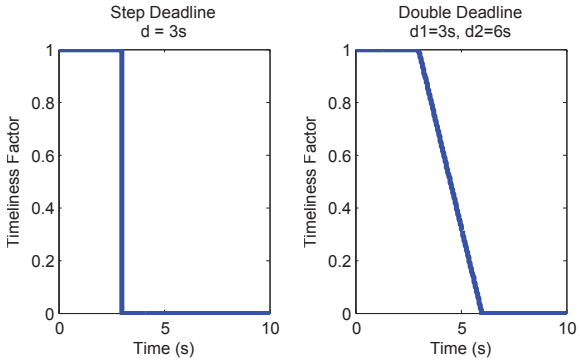

Figure 1: Example timeliness functions: step (left) and double deadline (right)

The timeliness attribute of QoI may be stochastic because transformations on data do not always produce deterministic results. For example, compression algorithms do not always achieve the same compression gain for different source files. Likewise, network performance is not always perfectly predictable.

2.2.2. Accuracy

The accuracy of information can be described as the degree to which the information correctly describes an event. This is an *intrinsic* attribute of the information because it does not depend on its use.

In computer vision applications, accuracy represents the ability of the application to correctly interpret the image. In biometric matching applications, the accuracy will be the probability that the biometric may be correctly matched to a person. For OCR applications it will be the probability that words are correctly extracted from an image.

The accuracy of information is affected by both network conditions and data attributes. A communications channel may insert errors into a data stream or cause parts of the data to be lost. These effects will lower the accuracy of the data, which will in turn negatively impact the accuracy of information derived from the data. To overcome these errors and losses, error recovery techniques such as forward error correction (FEC) or retransmission protocols may be used. However as noted above, these will affect the timeliness of the information. Likewise, transformations on the data, for example compression or resizing, can also affect the accuracy of the information derived from the data.

Here we see that some data transformations, specifically compression and resolution, affect more than one information attribute. In fact, they may affect the information attributes in opposite ways – increasing the compression ratio (and decreasing resolution) may improve timeliness (depending on the processing time and network rate) but degrade accuracy. It is exactly these relationships that QoI-aware networking is concerned with.

2.2.3. Precision

The precision of the system can be described as follows: "the degree of reproducibility of measured values which may or may not be close to real world value"[1]. For example, when describing a scene of a fire, it may be *accurate* to state which building is on fire, but this wouldn't be very precise. The precision of the description will be improved as the report gets more specific; for example by stating "the second floor of the building" is on fire.

As with accuracy, the precision of the information resulting from an application depends on the attributes of the data. This is affected by the data source and any transformations on the data, including network transformations. For example, as data is filtered or compressed, precision may be sacrificed.

For image applications, precision will be instantiated by the detail distilled from the image. For example, if an image is used to count the people in a crowd, the precision will be defined by the granularity of the count. It might be accurate to state there are "more than 50 people" in the room, but if the precision can be improved, it might be possible to state there are "exactly 53 people" in the room.

The precision of data is usually affected indirectly, in an attempt to change some other attribute. For example, one way to improve timeliness is to reduce the size of the data being transferred by reducing the resolution of the data, and thus the precision.

2.2.4. Other Information Attributes

There are many other attributes of information. These include provenance, integrity, freshness, and completeness. Some of these may be affected by network transfer, processing performed on the information and data used to derive it, and the information sources. The interested reader is referred to [1, 2, 3] for a more complete discussion.

3. Experimental Setup

To quantify the tradeoffs between different data quality metrics and their impact on QoI, we chose OCR as a representative application. We use a dataset originally collected at University of Nevada Las Vegas (UNLS/ISRI annual test)[4]. The dataset includes 100 business letters scanned using 300 dpi for which we have the ground truth text. The images have a resolution of 1712x2200 and an average size of 366KB.

Initially, we used MATLAB simulations for our investigation. Later, in attempt to validate our findings, we ran the experiments on actual mobile devices and found that our simulations overlooked key considerations of real-world implementations. Nonetheless, we present our initial simulation data, and in a later section, along with our implementation data, we discuss our initial oversights and compare the two datasets where appropriate. Where the comparison between the simulation data and implementation data is interesting, we discuss the data and highlight any differences.

Table 1: FEC schemes used in Simulation

Encoding	**Params**	**Overhead**	**Notes**
Hamming	(7,4)	75%	4 parity bits for each 3 bits of information
Reed Solomon	(255,223)	14%	223 bit symbols coded as 255 bits; $\lfloor(255 - 223)/2\rfloor$ correction ability
Reed Solomon	(255,251)	1.6%	251 bit symbols coded as 255 bits; $\lfloor(255 - 251)/2\rfloor$ correction ability
(No FEC)		0%	No correction ability

To assess QoI, we use two information metrics – timeliness and accuracy. For timeliness, we consider both a step function and double deadline function. Examples are shown in Figure 1.

3.1. Simulation

In order to study the tradeoff between timeliness and accuracy, for each document in the dataset, we tested the effects of two data reduction schemes – compression (JPEG, lossy) and resizing (resampling). For each scheme, we test the effects on error-free images, as well as images where we introduce uniformly distributed random bit errors at various error rates.

To increase information accuracy at the sink in the face of bit errors, we use four FEC schemes in our simulations. The details of the FEC schemes used in the simulation can be found in Table 1.

The tests and processing were done almost exclusively in MATLAB, with the exception of the OCR processing, which was done externally in Tesseract[1].

3.2. System Design

In order to assess our simulations and better analyze real-world conditions, we designed a general QoI query-response system comprised of two main parts, the Server and the Client. In our specific implementation, we used a computer for the Server and an Android mobile handset for the client. This system is depicted in Figure 2.

$$\textsc{Query} = \big[r, f, q\big] \tag{4}$$

[1] `http://code.google.com/p/tesseract-ocr/`

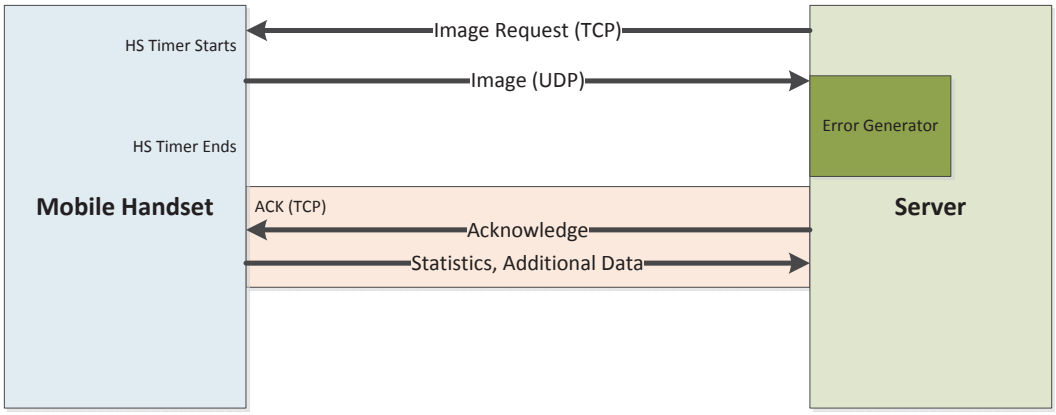

Figure 2: Server/Client implementation diagram

As illustrated by (4), the Server sends the Client a query, specifying a file request r, QoI function f, and QoI requirement q.

Upon receiving the query, the Client looks to see if it has a matching file, and if it does, evaluates the QoI function to determine, what, if any, data transformations should be applied prior to transmitting the file to the Server.

The two parts of the message, the QoI function and QoI requirement can be explained as follows. The **QoI function** instructs the Client how to calculate QoI values for a given candidate response while the **QoI requirement** is a minimum QoI value, calculated using the specified QoI function, that must be met by any response.

For example, a trivial QoI function might be equating QoI to image quality. In this case, the associated QoI requirement would just be an image quality requirement. Accordingly, for our OCR implementation we used the QoI function shown in (2).

Once the Client receives the QoI function and requirement (and a matching file is present) it estimates the QoI value for the candidate responses it is capable of sending (via the provided QoI function), selects the optimal candidate response, and sends it back to the Server.[2] It is possible that no matching file is present, or that a match is present but none of the candidate responses are expected to meet the QoI requirement, in both cases no file is returned; no response is sent.

This highlights the difference between requested QoI and delivered QoI:

- **Requested QoI** is the QoI requirement sent from the Server to the Client in order to specify a certain minimum threshold for responses.

[2]Provided that the estimated QoI for the optimal candidate response meets or exceeds the requested QoI

- **Estimated QoI** is the QoI value that the Client calculates (estimates) for each candidate response. The candidate response with the highest estimated QoI, provided it meets the requested QoI, is sent in response to the query.
- **Delivered QoI** is the QoI value of a response, as calculated by the Server (receiver).

The estimated QoI values are calculated locally by the Client, leveraging its knowledge of the local state (e.g. battery life, estimated processing time) and network state. Allowing the Clients to determine these values themselves eliminates the need to constantly exchange state information updates with a central decision maker. However, since these are only predictions, they may differ from the QoI calculated by the server upon receipt (received QoI) – for example, if the network state changes or a file takes unexpectedly long to process.

3.3. Implementation

In an effort to realize a system as described previously, we implemented a QoI-aware OCR query-response system with a Google/Samsung Nexus S Android mobile phone, and a computer.

In our implementation, the Server, a Python script, sends a JSON-encoded query over TCP to the handset. This query includes a filename pattern (e.g. `IMG_00042.jpg`, or `*.png`), a QoI function and a QoI requirement. Upon receiving and parsing the query, the Client first looks to see whether it has any matching files – if it does not, it discards the query and resumes waiting for connections.

If matching files are present, the Client estimates the QoI value for all of the candidate responses it is capable of sending. In our OCR application, this includes:

- No resizing, FEC
- No resizing, No FEC
- Resizing (75%), FEC
- Resizing (75%), No FEC
- Resizing (50%), FEC
- Resizing (50%), No FEC

Its important to note that during this candidate evaluation phase, no data transformations are actually being done – rather, the estimated QoI is being calculated. In the case of our implementation this is done by referencing curve-fit n-th (typically 2nd) order functions that predict the resulting processing time and output file size for each operation. By "daisy chaining" these estimations together, applying the output of one as the input of the next, you end up with an estimate of the final filesize, processing time and QoI.

After determining the estimated QoI for each candidate, the Client then selects the candidate response with the highest estimated QoI value and, if that value meets or exceeds the requested QoI, performs the specified transformations, and transmits it in response to the Server.

In our implementation this response was over UDP. The choice of TCP in the forward direction and UDP in the reverse were just implementation decisions.

Table 2: FEC schemes used in Implementation

Encoding	**Params**	**Overhead**	**Notes**
Hamming	(8,4)	100%, 103ms/kb	Hamming (7,4) with an extra parity bit to identify two-bit errors.
Reed Solomon	(15,11)	36%, 303ms/kb	Allows the receiver to correct up to 2 erroneous symbols per 15 symbol block (each block is 4 bits).

TCP in the forward direction provides the Server implicit acknowledgement that the query was received (which may be especially important in the case where Clients are mobile devices, possibly without connectivity). UDP was chosen as the reply protocol in order to eliminate the overhead (both in total data and in retransmission overhead) required by TCP. Using UDP (and disabling checksumming) also allows us to look at the effect of "transparent" bit errors.

The files, as provided, are TIFF files, however, because the current Android libraries lack the tools to manipulate TIFF files natively, we converted each image file (losslessly) to a PNG file and stored them on the phone's internal memory. Unlike TIFF, the Android libraries *do* provide functions to manipulate PNG files – namely resize them – and if a resizing was necessary (i.e., if the optimal candidate was resized), the Client would use these libraries to resize the source image appropriately.

Finally, if forward error correction (FEC) was necessary (i.e., if the optimal candidate included FEC), in order to allow for error correction once the file was received, the Client would perform the necessary encodings prior to sending the file. In this case, however, because the current Android libraries don't contain FEC libraries, we implemented both Hamming and Reed-Solomon Forward Error Correction encoding ourselves.[3]

As stated previously, we developed pure-Java implementations of both Hamming and Reed Solomon forward error correcting codes. The details of our implementations are listed in Table 2. The overhead is shown in both the payload size increase as well as the average processing time per kb of data.

[3]These implementations are in "pure Java". That is, they do not take advantage of various optimizations and performance increases that might be realized if one were to implement the algorithms in native code. While the use of other, more optimized libraries might shift our results in one direction or another, it does not affect the fact that the tradeoffs exist between competing goals, and our overall thesis that studying these tradeoffs is necessary for more intelligent, QoI-aware networks.

Once the Client is done transforming the data, it transmits the (possibly-resized, possibly-encoded) image to the Server. In order to simulate errors in the transmission channel, it is at this point that we have the option to intentionally insert uniformly distributed random bit errors at a specified rate. We do this so that we can vary the error rate of the pre-decoded file in order to study the effect of channel error rate on the resulting OCR accuracy.

The Server then performs the necessary decoding, error correcting, and decompressing operations, runs the image through OCR software (Tesseract) and compares the output to the known ground-truth text. This comparison produces two scores, a *completeness score* (the number of words in the original document that were reproduced in the OCR output) and a *correctness score* (the number of words in the OCR output that were in the original document). In order to simplify the rest of the evaluation, we average these two numbers to obtain a single score for the transmission, however we note that some applications may have some other method for determining the overall score. This average becomes our application score, S, as previously described in section 2.1.

4. Results

Our goal is to quantify the tradeoffs of different data transformations (e.g., compression) and network controls (e.g., error recovery) on QoI. We can characterize these tradeoffs in two ways - *(i)* what settings achieve the maximum QoI, and *(ii)* what are the minimum resources required to achieve a target QoI. The first assessment allows us to determine the best a system can do, effectively answering the question "Is it possible to get information of sufficient quality?" The second allows us to make smarter resource allocation decisions, while still meeting the requirements of our users.

4.1. Baseline Simulations

In these experiments we consider two factors at the information level - the accuracy of the information (for which we use our application score), and the timeliness. In our experiments timeliness is affected by *(i)* any data transformations (e.g. compression and resolution); *(ii)* the timeliness function (step and double deadline); *(iii)* the data transfer rate of the network; and *(iv)* protocol overhead. That is,

$$\begin{aligned} \textsc{Timeliness} &= T(t) \\ &= T(t_p + t_t) \end{aligned}$$

where t_p is the processing time and t_t is the transmission time, which can be generalized as,

$$t_t = \frac{\textsc{Response Size (bytes)}}{\textsc{Avg. Network Rate (Bps)}} \times \textsc{Proto. Overhead Ratio} \tag{5}$$

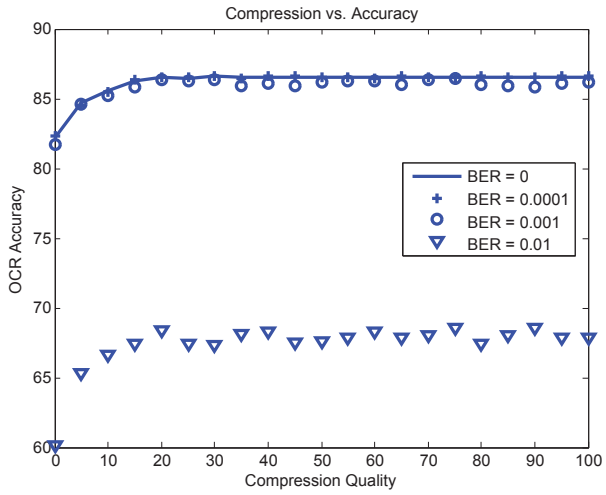

Figure 3: Effects of compression in OCR accuracy at different BERs – MATLAB simulation

The application score, S, is the OCR accuracy. With the accuracy and timeliness metrics determined, we can instantiate the QoI function as:

$$QoI := \text{ACCURACY} \times \text{TIMELINESS} \tag{6}$$

In this application a product of the two metrics suits our needs, however in general, QoI functions may be any arbitrary function of any number of terms.

Using both JPEG-compression and simple resizing, we calculated the QoI at various compression levels and resizing ratios. We did this both for perfect transmissions (error rate of 0) as well as for transmissions with error rates ranging from 10^{-2} to 10^{-4}. The results are shown in Figures 3 and 4.

There are a few points worth noting from these figures. First, the OCR results show that accuracy is relatively insensitive to compression quality. A JPEG-compression quality of 10 could reduce the file size by more than 75% without any significant impact on accuracy (<2%). On the other hand, the results show that accuracy *is* sensitive to resizing. A resizing scale of 0.5 would also reduce the file size by 75%, but in the case of resizing, the average accuracy dropped to less than 50%. We attribute this difference to the "intelligence" of the JPEG-codec and compression algorithm, making it much more effective as an accuracy-preserving data reduction tool. Third, the two figures show the effect of significant bit errors on OCR accuracy – infrequent errors are well-tolerated, but as the error rate approaches 0.01 and higher, the accuracy begins to be noticeably affected. Lastly, since these graphs were simulated, we weren't able to take into account processing time and simply assumed that the deadline was long enough to accomodate any processing and transmission of any size.

Now, we look at the effect of using various FEC schemes in an attempt to overcome a noisy transmission channel. For the following figures, we use a BER of 10^{-2}. First, we look at various compression factors – the results are shown

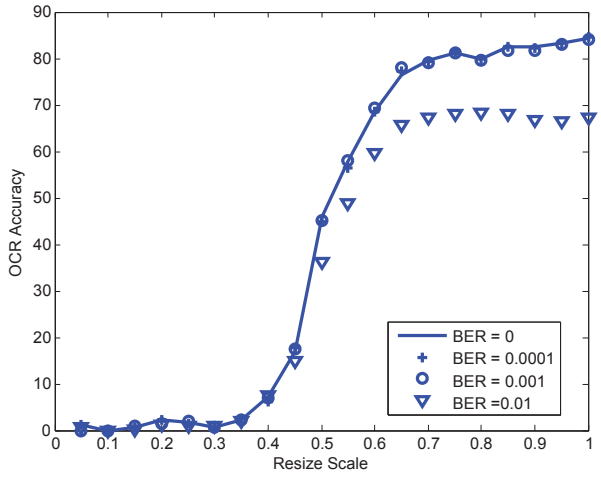

Figure 4: Effects of resizing in OCR accuracy at different BERs – MATLAB simulation

in Figure 5. Next, we look at the effect of using FEC for images of various resolutions (a result of resizing) – the results are shown in Figure 6.

As shown in these figures, and as would be expected, using FEC generally improves OCR accuracy. In fact, in all cases in our simulation, not using FEC was strictly worse than using FEC. Even the relatively "lightweight" Reed Solomon (255,251) improves accuracy, although only slightly. The more robust FEC schemes result in greater improvements, but, of at the cost of more overhead.

4.2. Implementation Results

These simulation results were generally consistent with our implementation results for the same tests.

Figure 7 shows the results of the same Accuracy vs. Resize Scale test (Figure 4), but actually implemented on a mobile device as opposed to simulated results. This figure also contains error bars around the mean, better illustrating the test instance variance (Figure 4 displays only the mean). Note that the curve for the two lowest error rates (0 and $1e^{-6}$) is extremely similar to the simulated curve in Figure 4.

Figure 8 shows the results of the same Accuracy vs. Resize Scale test (Figure 6). Both Figures show a noticeable drop in accuracy below a scale of 0.6, and both illustrate the benefit of adding FEC encoding when there is a significant error rate in the network. The main difference between the two figures is that the unencoded performance was significantly better in the simulation than in the actual implementation. We attribute this difference to the different resizing libraries used for the tests. In the simulation we used the JPEG resizing functions built into MATLAB (`imresize`) and in the implementation we used the Android built-in Bitmap functions.

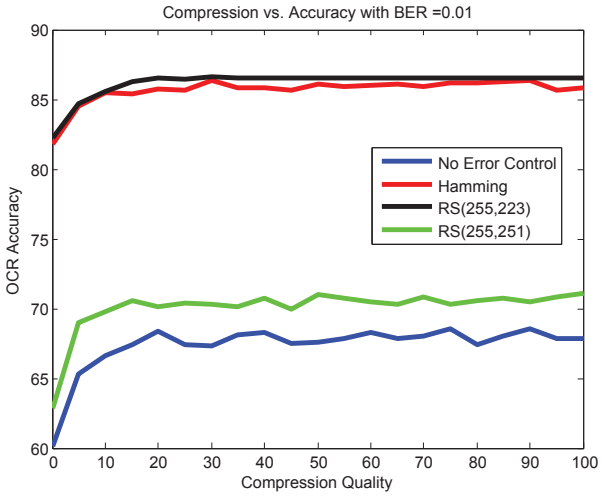

Figure 5: OCR Accuracy vs. Compression for BER = 10^{-2} – MATLAB simulation

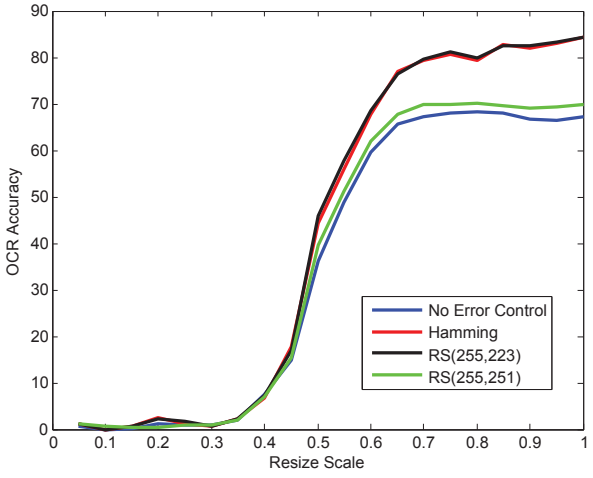

Figure 6: OCR Accuracy vs. Resizing for BER = 10^{-2} – MATLAB simulation

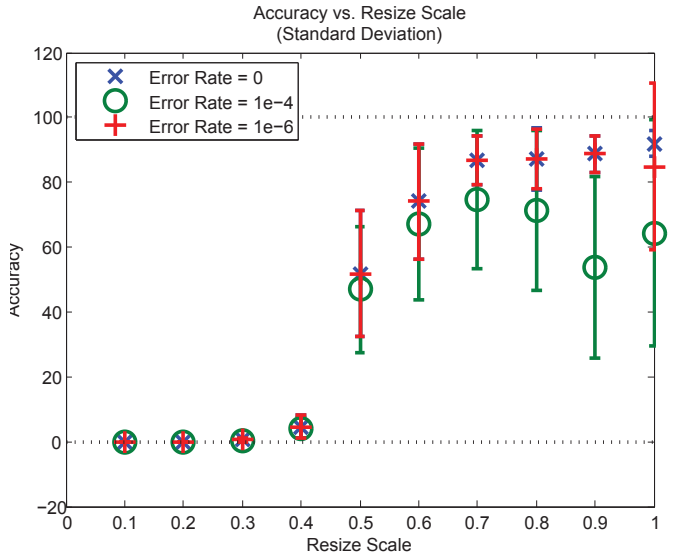

Figure 7: OCR Accuracy vs. Resize Scale at various BERs – Experimental

The other, minor difference is that the simulation data plots the "Accuracy" while the implementation data plots the "Achieved Accuracy". Due to OCR limitations, even a perfect transmission of the file would almost always result in an accuracy of less than 100%. This "baseline accuracy" could range anywhere from 80% to 100% and depended only on the original file. In order to make our tests file-agnostic, we define "Achieved Accuracy" as $\frac{\text{Test Accuracy}}{\text{Basline Accuracy}}$.

4.3. QoI and Timeliness

As previously discussed, one way to improve timeliness is to reduce the image resolution through resizing. Figures (9) and (10) show the impact on QoI when using resizing with a step function deadline and a double deadline function, respectively.

The two figures are similar up to a resize scale of around 0.7, and then begin to diverge for scales between 0.7 and 1.0. The reason for this is the timeliness function. In the first graph, using the step function deadline, QoI was not affected as long as it made the 150 packet deadline. In the second graph, using the double deadline, QoI *is* affected (negatively) for any images that exceed the 75 packet soft deadline. This difference in timeliness functions is what causes the second graph to have a steeper drop in QoI as you approach a scale of 1.0 (the largest of the filesizes, and largest of the transmission times).

Our implementation data shows similar trends. Figure 11 shows our implementation data for a similar test.

Here, however, we're using a time deadline instead of a deadline based on the number of packets. There are two differences between the graphs that are worth noting. The first, that the unencoded performance is significantly better

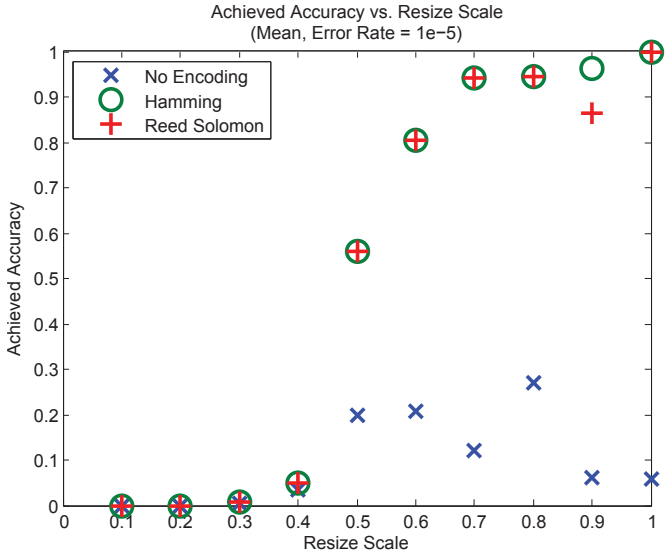

Figure 8: Effects of resizing in OCR accuracy at different BERs – Experimental

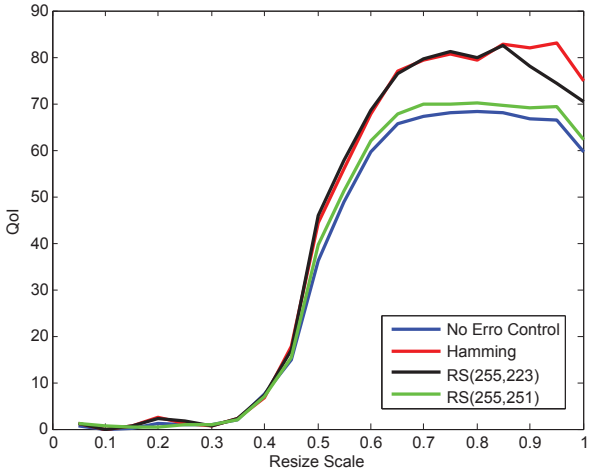

Figure 9: QoI with timeliness step function (d=150 pkts) for BER = 10^{-2} – MATLAB simulation

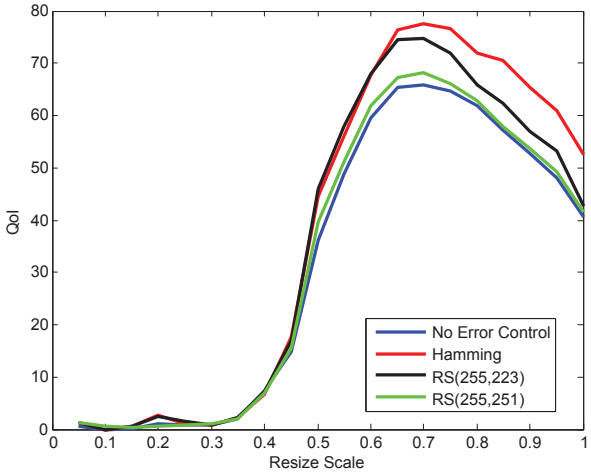

Figure 10: QoI with double deadline function (d_1=75 pkts, d_2=150 pkts) for for BER = 10^{-2} – MATLAB simulation

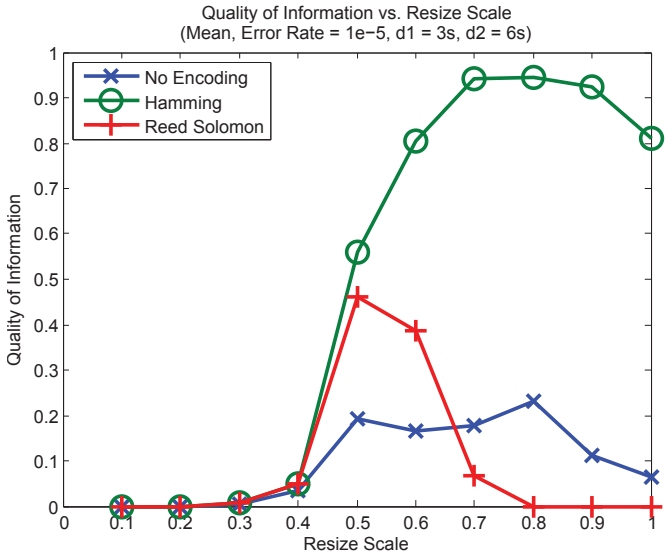

Figure 11: QoI vs. Resize Scale for various FEC schemes, double deadline – Experimental

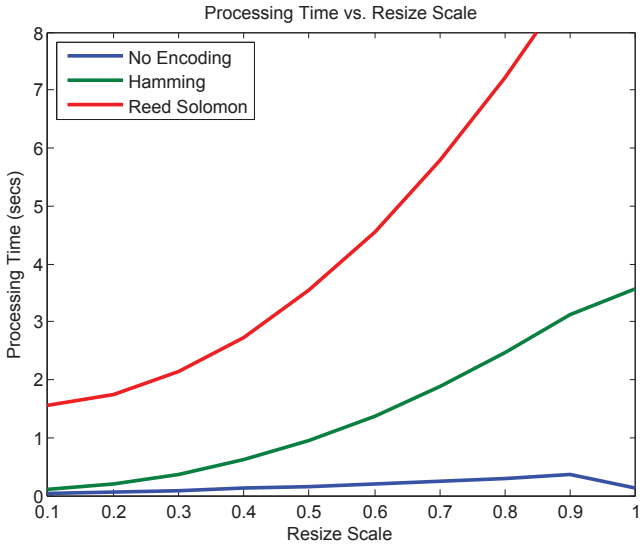

Figure 12: Processing time for various resize scales and encoding schemes

in the simulation than in the implementation, we've already discussed. The second, that the Reed Solomon encoding performance drops off significantly in the implementation at around a scale of 0.5 whereas in the simulation, no such drop is seen.

This was one of the interesting developments we noticed during our implementation. Why did the performance drop off so significantly in the implementation (even becoming worse than no FEC at all at high scale ratios) when no drop was seen in the simulation? The answer is actually very simple: Reed Solomon encoding on the phone takes a significant amount of time – often in the order of 10-15 seconds. This processing time was simply not accounted for in our simulations and greatly affects the timeliness factor of the QoI equation.

Figure 12 shows the estimated processing time for a given file, resize scale and encoding scheme.

As shown in the figure, Reed Solomon encoding takes a considerable amount of processing time, much more time than the other implemented encoding scheme (Hamming) and orders of magnitude more than using no encoding. This figure illustrates the cause of the difference between the simulated data in Figure 10 and the implementation data in Figure 11.

4.4. QoI-aware Network Optimization

The results presented so far give an indication of the highest QoI that can be achieved, and the settings to achieve it, given a network rate, error rate and class of source images. Another interesting question is: given a required QoI, what setting will meet the requirement using the smallest amount of network

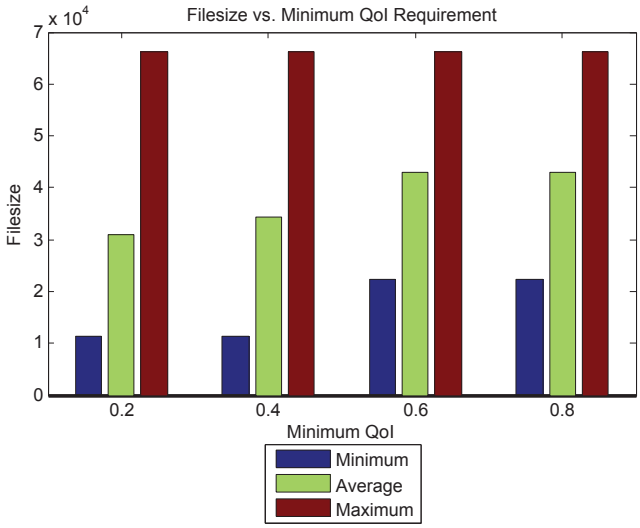

Figure 13: Minimum, average and maximum filesizes of candidate transformations of various minimum QoI requirements

resources. In fact, we can use the same data we've just presented to answer this question.

If we take, for example, Figure 11 and draw a horizontal line across the graph at a given QoI, we can see which schemes and resize scales will meet our QoI requirement. We can call these (schemes, resize scale) pairs our candidate transformations. For each candidate transformation, we can estimate the resulting filesize.

Figure 13 shows the minimum, average, and maximum filesizes for the candidate transformations for various minimum QoI requirements. Of note is the fact that this graph can be thought of as an extension of Figure 11 – once the candidate transformations are identified (eg. via Figure 11), a graph like this can be constructed.

Here we see that, if you were requesting a QoI of 0.8, you wouldn't gain much in terms of increased network availability if you lowered this to, say 0.6, because the minimum filesize candidate transformation is the same for the two QoI requirements. However, if you were requesting a QoI of 0.6, lowering it to 0.4 permit a smaller candidate, lowering your required network usage by 49%! In other words, if you are slightly flexible in your QoI requirement, you may be able to identify opportunities to further increase performance by slightly relaxing your conditions.

5. Related Works

In spite of the fact that QoI-aware networking is relatively new area, several efforts in various disciplines have studied QoI from various perspectives. Statisticians first studied data quality in the late 1960s [1]. The concept was then adopted in computer science for data mining and data warehousing.

Several metrics and measurements of information quality have been studied. Bouzeghoub and Peralta [5] proposed a framework for analyzing the data freshness in warehousing where data are not materialized in the real time. Others proposed grouping quality measures into intrinsic, relational or contextual metrics and reputational metrics [6]; we reuse these ideas here. Other studies focus on credibility [7, 8].

P. Missier et al. in 2006, [9] proposed quality views that incorporate user specifications in reusing quality components in software development. Although this study applies to subjective to genomics data, it can be helpful in building relationships between quality measures. For military sensor networks, a probabilistic approach has been proposed to estimate the QoI [10]. Sensor level fusion algorithms based on feedback from sensors have been studied by [11]. Their study contributes to optimizing resource allocation decision through QoI metrics in addition of fusion of multiple sensors to enhance the quality.

There has also been a great amount of work on QoI for specific applications. For example, there has been work done to characterize the quality of images via PSNR, MSE and other measures, but these do not rate the ability to distill information from the images. Likewise there has been a tremendous amount of work on the impact of compression on biometric matching. This enlightens us on one aspect of QoI, namely the accuracy versus compression.

6. Conclusion

In this work, we proposed a QoI-aware model for information delivery. We defined the concept of QoI functions and suggested one such function for a model application, Optical Character Recognition. We specifically considered timeliness and accuracy attributes of information, and showed how these attributes are impacted by attributes of the data from which the information is derived. We presented our initial findings obtained via simulation as well as those obtained via our actual implementation on mobile phones. When the data differed, we presented our thoughts on why and highlighted areas we overlooked during our initial simulations. Using the data obtained, we showed how processing time plays a significant, and previously underestimated role in the overall quality of the information.

Finally, we showed how adjustments to QoI requirement could have a significant impact on the resources required from the network, hopefully motivating the benefits of QoI-aware networks.

Future work is focused on studying different quality metrics such as completeness, freshness and provenance. Moreover, different error models such as

burst errors and node failure can be studied as well as how well this model can be applied to other media (video, audio, text).

We're currently looking into ways to make networks, in addition to indivial hosts, more QoI-aware, and preliminary results are encouraging.

References

[1] C. Batini, C. Cappiello, C. Francalanci, and A. Maurino, "Methodologies for data quality assessment and improvement," *ACM Comput. Surv.*, vol. 41, pp. 16:1–16:52, July 2009.

[2] V. Sachidananda, A. Khelil, and N. Suri, "Quality of information in wireless sensor networks: A survey," *In Proc. of The International Conference on Information Quality (ICIQ)*, vol. (Accepted, to appear), 2010.

[3] E. Gelenbe and L. Hey, "Quality of information: An empirical approach," in *Mobile Ad Hoc and Sensor Systems, 2008. MASS 2008. 5th IEEE International Conference on*, 29 2008-oct. 2 2008, pp. 730 –735.

[4] J. K. S. V. Rice and T. A. Nartker, "The third annual test of ocr accuracy," TR 94-03, ISRI, University of Neva- da, Las Vegas,, Tech. Rep., April, 1994.

[5] M. Bouzeghoub, "A framework for analysis of data freshness," in *Proceedings of the 2004 international workshop on Information quality in information systems*, ser. IQIS '04. New York, NY, USA: ACM, 2004, pp. 59–67.

[6] B. Stvilia, L. Gasser, M. B. Twidale, and L. C. Smith, "A framework for information quality assessment," *JASIST*, vol. 58, pp. 1720–1733, 2007.

[7] B. J. Fogg and H. Tseng, "The elements of computer credibility," in *Proceedings of the SIGCHI conference on Human factors in computing systems: the CHI is the limit*, ser. CHI '99. New York, NY, USA: ACM, 1999, pp. 80–87.

[8] D. McKnight and C. Kacmar, "Factors and effects on information credibility," *International Conference of Electronic Commerce. 2007*, pp. 423–432.

[9] P. Missier, S. Embury, M. Greenwood, A. Preece, and B. Jin, "Quality views: capturing and exploiting the user perspective on data quality," in *Proceedings of the 32nd international conference on Very large data bases*, ser. VLDB '06. VLDB Endowment, 2006, pp. 977–988.

[10] D. Gillies, D. Thornley, and C. Bisdikian, "Probabilistic approaches to estimating the quality of information in military sensor networks," *The Computer Journal*, 2009.

[11] Y. K. Y. C. Z.M. Charbiwala, S. Zahedi and M. Srivastava, "Toward quality of information aware rate control for sensor networks," *Fourth International Workshop on Feedback Control Implementation and Design in Computing Systems and Networks*, pp. –, 2009.