Network-Conscious GIF Image Transmission Over the Internet*

Paul D. Amer
Sami Iren
Gul E. Sezen
Phillip T. Conrad
Mason Taube
Armando Caro

Computer and Information Sciences Department
University of Delaware, Newark, DE 19716 USA
Email: {amer,iren,sezen,pconrad,taube,acaro}@cis.udel.edu
Phone: (302) 831-1944 Fax: (302) 831-8458

Abstract

Traditional image compression techniques seek the smallest possible file size for a given level of image quality. By contrast, network-conscious image compression techniques take into account the fact that a compressed image will be transmitted over a network that may lose and reorder packets. We describe *GIFNCa*, a network-conscious revision of the popular GIF89a standard.[1] As with GIF89a, GIFNCa compresses an image using LZW encoding[2], however GIFNCa does so using an Application Level Framing approach. The data is segmented into path MTU-size data units, each of which can be independently decompressed and displayed on its own. Under lossy network conditions, when used in combination with an unordered transport protocol, GIFNCa permits faster progressive display at the receiver than GIF89a over an ordered transport protocol. This advantage comes in exchange for a small penalty in overall compression. This paper defines GIFNCa, and presents preliminary experimental data concerning this tradeoff. The overall goal of this research is to illustrate (1) the value of considering network characteristics in designing image formats, and (2) the value of *unordered* transport service.

Keywords: application level framing, GIF, image compression, Internet, multimedia, transport protocols

1 Introduction

Although the GIF89a compression standard [8] is intended for the on-line transmission and interchange of images, the file format is defined with the assumption that the channel between the storage of the file and the display of the image is totally ordered and reliable (i.e., no bit errors, no loss, no duplicates). GIF89a's format makes no provision for error-detection or error-correction. For decoding to occur at the receiver, ordered and error-free delivery of the GIF89a structure is required. Decompression and progressive display of image data that arrives out-of-order must be delayed until the missing data arrive.

*Prepared through collaborative participation in the Adv Telecomm/Info Dist'n Research Program (ATIRP) Consortium sponsored by ARL under Fed Lab Program, Cooperative Agreement DAAL01-96-2-0002. Also supported, by ARO (DAAL03-92-G-0070, DAAH04-94-G-0093). 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 ARL or the US Government.

[1]GIF89a is a Service Mark of CompuServe, Inc., Columbus, OH.

[2]LZW is patented by Unisys Corp.

GIF89a's intolerance to channel loss and reordering is reasonable when the channel is a local system's backplane (see Figure 1) or a dedicated circuit-switched phone line (see Figure 2). But presuming an underlying reliable, ordered channel is a costly assumption when the channel is the Internet. In this case, the foundation channel is the IP network layer protocol [14] which is inherently unreliable; packets are regularly lost and misordered.

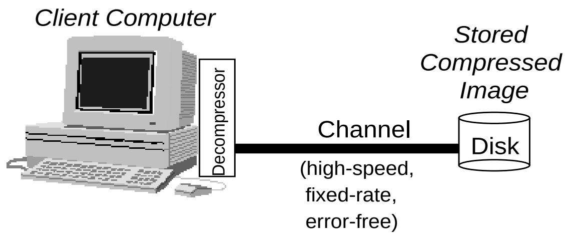

Figure 1: Local System Image Retrieval

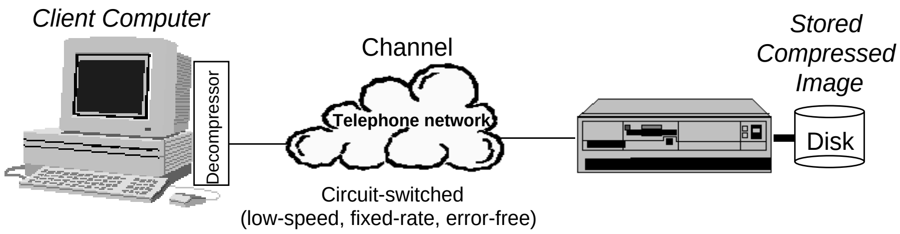

Figure 2: Image Retrieval Over a Dedicated Link

Applications such as Web browsers that communicate GIF89a images over the Internet's unreliable service are required to use TCP [15] on top of IP. TCP enhances the unreliable IP network service into a totally reliable, ordered transport connection (see Figure 3), but at the cost of extra delay and throughput.

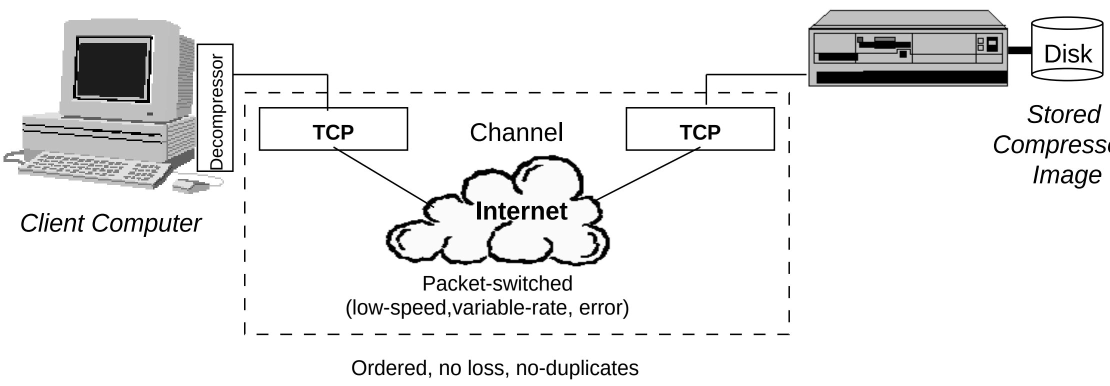

Figure 3: Image Retrieval over Internet via TCP

In this work, we apply the concept of network-consciousness [6], initially introduced by INRIA—Sophia Antipolis, to image compression, resulting in a variation of the GIF89a file format called *Network-Conscious GIF (GIFNCa)*. Network-conscious image compression [12] focuses not simply on maximizing compression; it focuses on optimizing overall progressive display performance when compressed images are transmitted over lossy packet-switched networks such as the Internet or battlefield networks. Faster progressive display is motivated for time-critical applications, e.g., a military target recognition system which may have to determine friend or foe before firing on a target. Likewise, faster progressive display is useful in situations where timely partial figures are desirable (e.g., users browsing the Web or advertisers wanting their publicity to appear as soon as possible to viewers).

The tradeoff between GIFNCa and GIF89a is one of compression vs. progressive display performance. GIF89a's advantage is its expected better compression. GIFNCa's advantage is its expected faster progressive display at the receiver.

The principle behind network-conscious image compression is Application Level Framing [5]. At the application layer, an image is divided into units no larger than the connection's path Maximum Transmission Unit (MTU)[3]. Each unit is independent and "carries its semantics". Therefore each unit can be delivered to the receiving application out-of-order to be immediately decompressed and displayed, thereby enabling faster progressive image display. An excellent description of the issues of out-of-order processing and some general simulation and experimental results can be found in [7].

Depending on whether or not an application can tolerate loss, there are two cases. Case (1) exists when the application must eventually receive the entire image without loss. Here the communication channel between the compressed GIFNCa file and the display must be reliable, although not necessarily ordered. Hence GIFNCa images can be transported across the Internet with transport services that offer an unordered reliable service rather than an ordered service (see Figure 4). Unordered protocols can offer shorter delays than ordered protocols such as TCP [2]. Case (2) exists when the application can tolerate some image loss. Then the communication channel can be unordered and either unreliable (e.g., UDP/IP), or partially reliable [4, 9, 13]. In this paper, we focus on case (1).

We acknowledge that there are significant barriers to the acceptance of GIFNCa as a standard, including the huge installed base of GIF87a and GIF89a images, and the fact that a reliable unordered transport protocol is required to achieve the full benefit of GIFNCa, to say nothing of the various proprietary and patent issues associated with GIF and LZW encoding. However, *the purpose of this research is not to promote a new image format, but rather to show the value of designing image compression algorithms with transport service in mind.*

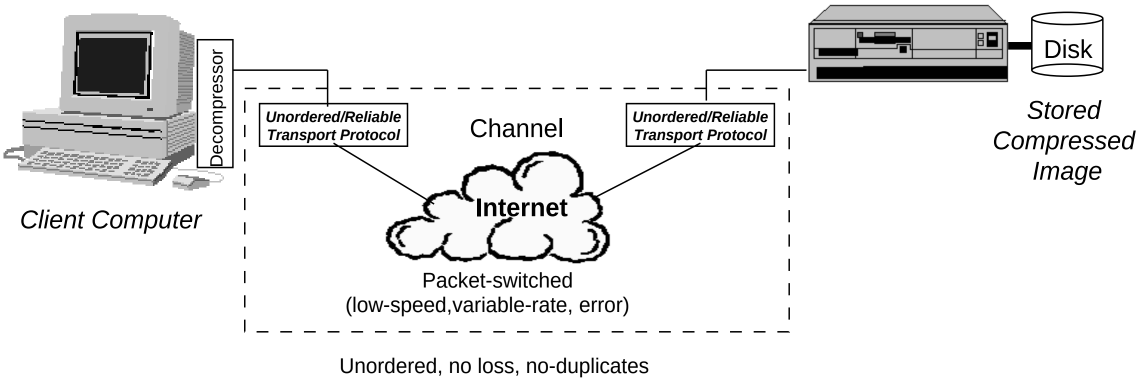

Figure 4: Image Retrieval over Internet via a Unordered, Reliable Transport Protocol

The remainder of the paper is structured as follows. Section 2 details the differences between GIF89a and GIFNCa so the reader can understand what changes are needed to allow segmentation into independent units. A few new fields were added to GIF89a; a few were modified. Section 3 presents experimental results demonstrating the advantages and disadvantages of GIFNCa vs. GIF89a. These preliminary data support the claim that as the underlying network loss rate increases, GIFNCa offers faster progressive display. General conclusions and future work are summarized in Section 4.

[3]MTU is the maximum frame size that a link layer can carry. A path MTU-size ADU is one that can be transmitted end-to-end without the need for IP layer fragmentation and reassembly.

2 Network-Conscious GIF

As stated in the Section 1, we propose an alternative to the GIF89a standard that is network-conscious. The result, GIFNCa, removes the ordered delivery requirement (and for some applications the reliability requirement) of GIF89a by framing image data at the compression phase (i.e., application level). The actual frame size is dictated by the path MTU size of the connection over which the image is expected to travel. Hence, the expected path MTU size must be known prior to compression, or an image must be compressed and stored according to multiple possible path MTU sizes.[4]

The GIF89a and GIFNCa file structures are shown in Figure 7. In GIF89a, most header information appears once. The GIFNCa structure, however, is broken into Application Data Units (ADUs). Each ADU containing image data also must contain enough header information to allow it to be decompressed at the receiving application (decoder) independent of the other ADUs. Thus, each ADU carries sufficient (possibly redundant) header information so the decoder knows how to process the ADU and where in the overall image the ADU's data should be displayed.

In defining GIFNCa, we try to remain loyal to the GIF89a specification as much as possible. We retain GIF89a's structure and location of fields wherever possible. To achieve a fair comparison, it is our goal that any differences between GIFNCa and GIF89a should be the result of making GIF89a network-conscious, and not by improving it in other ways. Our initial specification of GIFNCa in Figure 7 does not include GIF89a extensions. Including extension capabilities is reserved for future study, and should not impact upon the GIF89a vs. GIFNCa analysis in Section 3.

The need to redundantly include header information in multiple ADUs can result in lower overall compression. We argue that for some applications, the faster progressive display made possible because of unordered delivery compensates for this reduced compression. This tradeoff is described in Section 3.

We now distinguish between GIF89a and GIFNCa focusing on those components that needed to change to become network-conscious. The reader uninterested in these specific differences can skip to Section 3.

2.1 GIFNCa Field Descriptions

Some level of detail is necessary to fully understand what issues were confronted when transforming the existing GIF89a standard into a network-conscious one. Additional details on GIF89a can be found in [8]. In the following writeup, a prefix of [NEW] indicates a field introduced from the GIF89a version. A prefix of [MODIFIED] indicates a change.

Essentially, there are two types of ADUs that comprise a GIFNCa image: (1) color map ADUs, and (2) data ADUs. A color map ADU contains a GIFNCa signature, a screen descriptor, and a color map. A data ADU consists of: a GIFNCa signature, a screen descriptor, an image descriptor, position field, and raster data.

2.1.1 Signature

[MODIFIED] The GIFNCa *Signature* is always the ASCII string: `GIFNCa`. The first three characters identify the GIF type of compression and the last three characters stand for "Network-Conscious Version a". GIF89a's signature string only appears once. GIFNCa's signature must appear in every

[4] If an image is GIFNCa-compressed assuming a path MTU larger than that of the network over which it is eventually transmitted, fragmentation and reassembly will result. The overall system will operate correctly, but the expected performance gain from using GIFNCa will be reduced.

ADU. This is the first, albeit small, example of GIFNCa's redundancy that can lead to reduced compression.

2.1.2 Screen Descriptor

The *Screen Descriptor*, also called the logical screen descriptor, describes the overall parameters for all of the possibly multiple images in an image file. It defines the dimensions of the logical screen required, background screen color, color depth information, pixel aspect ratio, etc. It also identifies the type of ADU, that is, color map or data. GIFNCa extends GIF89a's *Screen Descriptor* by three bytes to a total of eleven bytes (see Figure 7).

Bytes 1-4 define the logical screen's width and height dimensions in pixels.

[MODIFIED] Then a 1-bit M flag identifies whether this ADU contains a color map or image data. GIF89a uses this bit to indicate whether or not there exists a global map. GIFNCa uses it to differentiate between ADU types. The distinction between local and global color maps are made by looking at the [NEW] 1-byte *Identifier* field as follows:

- if M=0, this ADU contains image data and *Identifier* contains an image number (1-255).
- if M=1 and *Identifier*=0, this ADU contains a global color map.
- if M=1 and *Identifier*>0, this ADU contains a local color map ADU that belongs to image number *Identifier*.

The 3-bit *cr* field represents the number of bits per primary color available to the original image, minus 1. For example, if the value is 3, then the original image had 4 bits per primary color available to create the GIFNCa image. Note that this does not mean that the GIFNCa image has 4 bits per primary color. The *cr* field only represents the richness of the palette from which the GIFNCa image was created. The GIFNCa image may be using only a subset of these colors.

The 1-bit S flag, if set, indicates that the local or global color map is sorted in order of decreasing frequency, with most frequent color first. This assists a decoder with fewer available colors in choosing the best subset of colors.

The 3-bit *pixel* specifies an exponent to calculate the number of bytes contained in the local or global color table (i.e., $2^{(pixel+1)}$. For example, if $pixel = 6$, then there are 128 colors.

The 8-bit *Background Color* specifies an index into the global color table indicating the default background color to be used for those pixels not covered by any image. If the ADU is not a global color map ADU, this field should be zero and ignored by a decoder.

The 8-bit *Pixel Aspect Ratio* is used to compute an approximation of the pixel aspect ratio in the original image. The aspect ratio is the quotient of the pixel's width over its height. If the value in this field is zero, it is ignored. Otherwise $Aspect\ Ratio = (Pixel\ Aspect\ Ratio + 15)\ /\ 64$. The 8-bit value range allows specification of the widest pixel of $4:1$ to the tallest pixel of $1:4$ in increments of $1/64^{th}$.

[NEW] If the ADU is a color map, the *X1,X2* bytes indicate the color map index start and end, respectively. Otherwise, the ADU contains raster data, and these two bytes represent the size of the ADU's raster data (in octets with *X1* representing the low order bits.)

At the end, one byte is reserved for future use.

2.1.3 Image Descriptor

An *Image Descriptor* specifies how to place an image on a logical screen. Each image has a unique image descriptor which must be repeated in each of the ADUs that belong to that particular image.

Without this redundancy, the receiver could not decompress the ADU independently.

Bytes 1-4 represent the left-top pixel coordinates within the logical screen. Bytes 5-8 represent the image's pixel width and height.

The 1-bit C flag identifies which color table to use: 0 for global; 1 for local.

[MODIFIED] The 1-bit I flag indicates if the image is interlaced. Since the primary motivation for GIFNCa is to obtain faster progressive display, this flag is always set.

[NEW] The 1-bit L and G flags help signal the end of an image and an image file respectively. The L bit is set for the last ADU of an image. The G bit is set for all ADUs of the last image in an image file. In GIF89a, a ';' defines the end of transmission. In GIFNCa, no analogous marker is possible since ADUs can arrive out of order. The L and G bits help the receiver determine when the entire GIFNCa structure has been received.

2.1.4 Position

[NEW] The *Position* in a data ADU specifies the pixel location from which the compression starts in that particular ADU. Since GIFNCa tries to eliminate network layer fragmentation/reassembly, it restricts the ADU size to the path MTU. Therefore, the LZW compression [19, 20] on a single image within an image file must be interrupted at some point not to exceed this maximum ADU size. The subsequent data ADUs will contain *positions* indicating their respective starting points.

2.1.5 Global/Local Color Map

Color Maps (or tables) are sequences of bytes representing red-green-blue color triplets. There can be at most one active color map for an image whose raster data field contains indexes into the active color map. The active color map can be either global to the whole image file or local to a single image within the image file. Both global and local color maps are optional.

There can be at most one global color map for an image file and at most one local color map for each image in an image file. Each color index points to a three-byte field which contains the r-g-b intensity levels, respectively. The color map size is calculated by using the *pixel* field in the screen descriptor. The size is equal to $3 * 2^{(pixel+1)}$.

2.1.6 Raster Data

[MODIFIED] The *Raster Data* component of a data ADU has the same structure as in GIF89a. However GIFNCa requires the first sub-block always to begin with an LZW clear code and the last sub-block always to end with an LZW terminator code. This represents an important *compatibility* between GIFNCa and GIF89a; the same software/hardware can be used for each one's LZW decompression.

The first byte of the *Raster Data* field contains the *LZW Code Size* which indicates the minimum number of bits required to represent the set of actual pixel values. Typically, this will be the same as the number of color bits (*pixel* field in the screen descriptor) used for this image. However, for black and white images (which only require one color bit), this value is set to two due to algorithmic constraints.

The *LZW Code Size* field is followed by one or more sub-blocks each of which consists of a 1-byte *Block Size* and 1-255 data bytes. The *Raster Data* of each ADU ends with a sub-block of size zero unlike in GIF89a where only one such zero-size sub-block appears at the end of the entire sequence of data sub-blocks.

3 Experiments

In this section we present a comparison of GIFNCa and GIF89a in terms of both absolute compression, and progressive display.

Given that GIFNCa compresses data only within an ADU rather than across ADUs, GIFNCa pays a performance penalty vs. GIF89a in terms of raw compression. Section 3.1 explores this penalty.

Turning to the measurement of progressive display, one faces a dilemma. If one uses the most advanced transport protocol available, namely TCP, one is forced to use ordered delivery. But in this context, GIFNCa offers no advantage over GIF89a. On the other hand, there is no commonly accepted unordered reliable protocol that features TCP-compatible congestion avoidance. As such, a direct comparison of an experimental unordered reliable protocol (without congestion control) vs. TCP (with its congestion control) is an unfair comparison. Our initial approach is to compare GIF89a vs. GIFNCa over two experimental transport protocols that differ only in that one is *ordered*, and the other is *unordered*. Both are reliable, and neither performs TCP-compatible congestion control. This allows us to focus on the primary issue of how network-consciousness combined with transport ordering affect the communication of images. In Section 4, we discuss future experiments that will take into account TCP-compatible congestion control.

3.1 GIFNCa's Compression Disadvantage

Past experiments show that the highest efficiency point for LZW compression is around 6K of compressed data [19]. Blocks smaller than 6K are penalized by LZW's start-up overhead. Blocks larger than 6K suffer a loss of efficiency because they lack stable statistics.

Since a typical path MTU size is usually 500-1500 bytes, GIFNCa is expected to result in a lower compression ratio. To roughly estimate the difference, we compressed nine images using both GIF89a and GIFNCa. These images were arbitrarily chosen with an attempt to include a number of military images since GIFNCa is under consideration for battlefield application. In all cases, compression values are for interlaced compression. The results are summarized in Table 1.

For example, a gray-scale tank image originally 256X256 pixels (i.e., 64K in uncompressed form) was compressed as an interlaced GIF89a file to 42.5K, a compression ratio of 1.50.

Using GIFNCa, the same tank image was compressed to 56.5K, 50K, and 45K for the three path MTU sizes 292 bytes, 576 bytes and 1500 bytes, respectively. These path MTU sizes are typical for: PPP links, the minimum size required for all Internet hosts [1], and Ethernets, respectively. The GIFNCa compression ratios are 1.13, 1.29, and 1.41, respectively. These ratios are 24.7%, 14.1%, and 6.1% lower (i.e., worse) than the compression achieved using GIF89a. Similar measurements are presented in Table 1 for the other eight images.

For the images tested and three path MTU sizes, GIFNCa resulted in an average compression roughly 25.4%, 14.6%, and 6.7% lower than GIF89a, respectively. In the worst case, GIFNCa resulted in 39.1% lower compression than GIF89a. This was for a color road map image and 292 byte path MTU size. In the best case, GIFNCa's compression was only 2.3% lower than GIF89a's. This occurred for a gray-scale aircraft image and 1500 byte path MTU size.

As expected the worst compression difference consistently occurs for the smallest path MTU size. In all cases, as the path MTU size increases, GIFNCa's compression improves both in absolute terms and relative to GIF89a.

Image	*Type*	*Original Size*	*GIF89a*	*GIFNCa (MTU size)* 292	576	1500
Tank	Gray	256x256 (64.0K)	42.5K (1.50)	56.5K (1.13) 24.7%	49.5 (1.29) 14.1%	45.3K (1.41) 6.1%
Aircraft	Gray	256x256 (64.0K)	38.8K (1.65)	52.6K (1.22) 26.1%	44.4K (1.44) 12.5%	39.8 (1.61) 2.3%
Lena 1	Gray	512x512 (256.0K)	236.7K (1.08)	298.6K (0.86) 20.7%	280.9K (0.91) 15.7%	258.3K (0.99) 8.3%
Lena 2	Gray	256x256 (64.0K)	68.0K (0.94)	79.8 (0.80) 14.7%	74.0K (0.86) 8.1%	70.9K (0.90) 4.0%
Cartoon	Color	530x400 (207.0K)	103.4K (2.00)	149.8 (1.38) 31.0%	129.2K (1.60) 20.0%	113.3K (1.83) 8.8%
Balloon	Color	150x166 (24.3K)	14.4K (1.67)	17.8K (1.35) 18.9%	15.4K (1.56) 6.5%	14.9K (1.61) 3.6%
Map	Color	480x489 (229.2K)	43.8K (5.23)	72.0K (3.18) 39.1%	57.9K (3.96) 24.3%	49.6 (4.62) 11.6%
Poppies	Color	148x274 (39.6K)	4.2K (9.4)	6.2K (6.27) 33.2%	5.2K (7.5) 20.2%	4.7K (8.37) 10.9%
Globe	Color	140x140 (19.1K)	13.8K (1.38)	17.3K (1.10) 20.4%	15.4K (1.24) 10.2%	14.5K (1.31) 4.8%

Table 1: Compressed File Sizes (and Ratios) for GIF89a vs. GIFNCa

3.2 GIFNCa's Progressive Display Advantage

We ran a set of experiments comparing GIF89a over a reliable ordered transport protocol called *Sequenced Protocol (SP)* vs. GIFNCa over a reliable unordered protocol called *Xport Protocol (XP)*. SP and XP were both developed as part of the *Universal Transport Library* at the University of Delaware [3, 2]. Both SP and XP are implemented at the user-level over UDP, and use the same code for all functions (including such functions as connection establishment/tear-down, round-trip-time estimation, retransmission timeout, acknowledgments, etc.); the only difference is that SP provides packet resequencing (i.e., ordered service) at the receiver, while XP does not.

Each experiment downloads a compressed image from server to client using an interface similar to familiar web browsers (see Figure 5). Packets are routed through a *Lossy Router*, a modified IP router that can simulate any of three loss models (Bernoulli, burst (2-Step Markov), or deterministic), and a *Reflector* that delays forwarding of IP packets to simulate lower bandwidth links. In the near future, we plan to replace the Reflector with a low bandwidth combat net radio SINCGARS or wireless link. In all experiments repeated, the Reflector simulated a 28.8 Kbps link. Future experiments will investigate other speeds.

First, we performed 20 experiments using the aircraft image with 0% IP packet loss: 10 each for GIF89a and GIFNCa. The results of these experiments are summarized in Table 2. For example, in experiment number 1, with GIF89a the user is able to see 24%, 46%, 72%, 92%, and 100% of the image data at times 3, 6, 9, 12, and 15 seconds respectively. With GIFNCa the user sees 21%, 41%, 64%, 81%, and 100% of the image at the same respective time periods. Table 2 clearly shows that GIF89a outperforms GIFNCa when there is no network loss or reordering. The reason is GIFNCa's

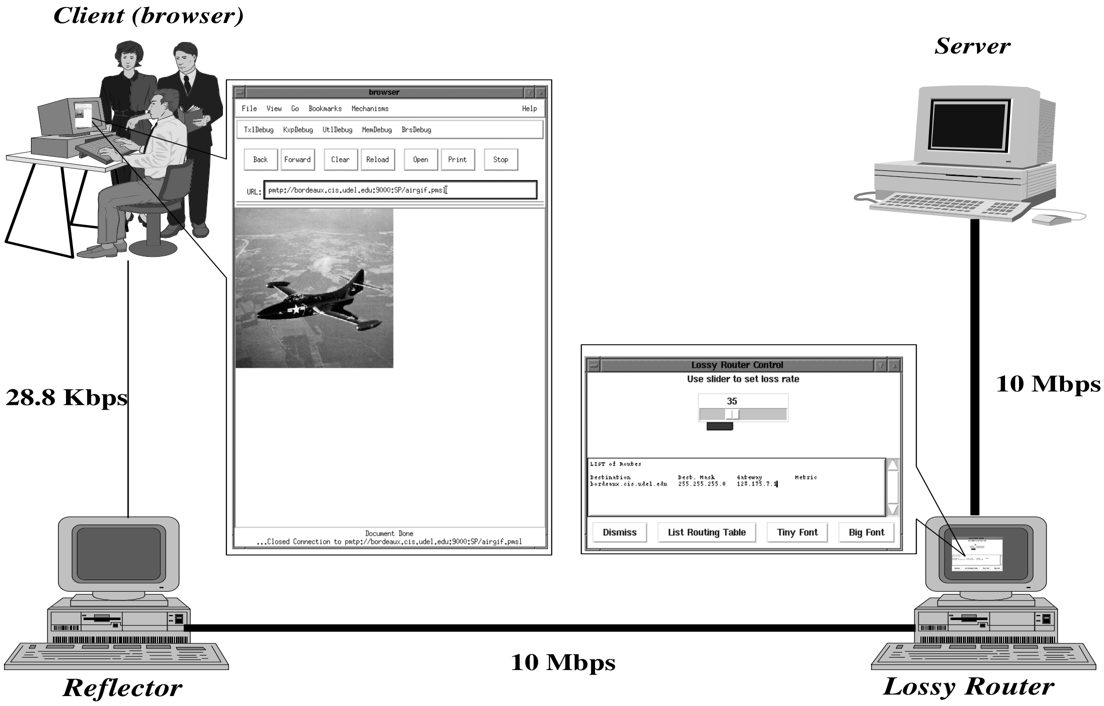

Figure 5: Testing Environment

compression disadvantage as explained in Section 3.1.

	GIF89a					*GIFNCa*				
	Time (sec)					*Time (sec)*				
Experiment no	*3*	*6*	*9*	*12*	*15*	*3*	*6*	*9*	*12*	*15*
1	24	46	72	92	100	21	41	64	81	100
2	21	43	69	90	100	22	41	64	81	100
3	21	45	70	91	100	21	41	64	81	100
4	22	45	70	91	100	21	41	64	80	100
.			.					.		
10	22	45	70	91	100	22	42	65	82	100
Avg	22	45	70	91	100	21	41	64	81	100
stdev	1	1	1	1	0	0	0	0	0	0
median	22	45	70	91	100	21	41	64	81	100

Table 2: Percentage of the Aircraft Image Being Displayed at Various Times (0% Loss)

Then, we performed a set of 120 experiments: 20 each for GIF89a and GIFNCa, at three different loss rates: 5%, 10%, and 15%. The results of these experiments are summarized in Tables 3, 4, and 5.

Unlike Table 2, Tables 3, 4, 5 show that GIFNCa outperforms GIF89a in terms of faster progressive display even at a loss rate as low as 5%. Table 3 shows that at 5% loss, with GIF89a on the average 16%, 57%, 95%, 99%, and 100% of the image is displayed at times 5, 10, 15, 20, and 25 seconds respectively. With GIFNCa, on the other hand, the percentages being displayed at the analogous times are 34%, 67%, 96%, 100%, and 100%. This means that after 5 seconds into the transmission, with GIF89a only 16% of the total image data is displayed. Whereas, with GIFNCa, this amount is 34%. Similarly, after 10 seconds, these numbers are 57% for GIF89a and 67% for GIFNCa. And so on.

The averages in Tables 2, 3, 4, and 5 are graphed in Figure 6. These graphs show that the GIFNCa

	GIF89a							*GIFNCa*						
	Time (sec)							*Time (sec)*						
Experiment no	*5*	*10*	*15*	*20*	*25*	*30*	*35*	*5*	*10*	*15*	*20*	*25*	*30*	*35*
1	16	49	95	100	100	100	100	35	69	98	100	100	100	100
2	29	63	100	100	100	100	100	36	70	100	100	100	100	100
3	8	41	100	100	100	100	100	35	70	98	100	100	100	100
4	17	76	94	100	100	100	100	36	70	100	100	100	100	100
.	.							.						
20	12	12	97	100	100	100	100	34	68	98	100	100	100	100
Avg	16	57	95	99	100	100	100	34	67	96	100	100	100	100
stdev	16	20	17	3	0	0	0	9	10	9	1	0	0	0
median	14	62	100	100	100	100	100	36	70	98	100	100	100	100

Table 3: Percentage of the Aircraft Image Being Displayed at Various Times (5% Loss)

	GIF89a							*GIFNCa*						
	Time (sec)							*Time (sec)*						
Experiment no	*5*	*10*	*15*	*20*	*25*	*30*	*35*	*5*	*10*	*15*	*20*	*25*	*30*	*35*
1	7	16	100	100	100	100	100	36	69	97	100	100	100	100
2	16	63	100	100	100	100	100	36	69	97	98	98	100	100
3	0	62	100	100	100	100	100	35	69	92	100	100	100	100
4	0	46	46	85	100	100	100	36	68	98	100	100	100	100
.	.							.						
20	18	66	88	88	100	100	100	35	69	98	100	100	100	100
Avg	2	40	73	90	97	99	99	34	68	96	99	99	100	100
stdev	6	24	31	19	7	4	4	3	4	4	1	1	0	0
median	0	47	89	100	100	100	100	35	69	97	100	100	100	100

Table 4: Percentage of the Aircraft Image Being Displayed at Various Times (10% Loss)

performance increases *relative to* GIF89a as the loss rate increases. This result is intuitive because as the loss rate increases so does the number of buffered out-of-order packets waiting for missing packets (in the case of GIF89a over ordered transport protocol). On the other hand, an unordered transport protocol (in the case of GIFNCa) delivers these out-of-order packets to the application (browser) as soon as possible after they arrive; no buffering for reordering purposes is needed.

To appreciate the significance of these numbers, Figures 8, 9 and 10 show actual images that the application displayed at 5, 10, 15 and 20 seconds for the most "typical" runs for each loss rate; that is, the runs that are closest in mean-squared distance to the averages over all experiments within a group.

To better assess the subjective value of partial images, consider three objectives that the user requesting the image may have:

1. to identify the image, that is, identify it is an airplane,
2. to identify whether the image represents a friendly or enemy target by identifying the insignia,
3. to identify the background details.

For objective one, with GIFNCa at 5 seconds, regardless of loss rate, the user can distinguish an aircraft. With GIF89a, on the other hand, it takes 10, 10, and 15 seconds to distinguish an aircraft at 5%, 10%, and 15% loss rates, respectively.

For objective two, at 10 seconds, the insignia is distinguishable at 5% and 10% loss rates with both GIF89a and GIFNCa. At 10 seconds and 15% loss rate, however, with GIF89a, the insignia

	GIF89a							GIFNCa						
	Time (sec)							Time (sec)						
Experiment no	*5*	*10*	*15*	*20*	*25*	*30*	*35*	*5*	*10*	*15*	*20*	*25*	*30*	*35*
1	13	13	42	100	100	100	100	34	67	94	100	100	100	100
2	1	49	82	100	100	100	100	35	69	97	100	100	100	100
3	0	7	12	86	100	100	100	35	69	95	98	100	100	100
4	7	22	22	80	100	100	100	34	70	100	100	100	100	100
.	.							.						
20	4	13	62	100	100	100	100	35	69	98	100	100	100	100
Avg	4	27	61	90	93	96	100	34	67	94	99	100	100	100
stdev	5	15	30	15	15	14	0	3	7	8	2	0	0	0
median	4	21	74	100	100	100	100	35	69	96	100	100	100	100

Table 5: Percentage of the Aircraft Image Being Displayed at Various Times (15% Loss)

is unrecognizable but is beginning to show itself with GIFNCa. At 15 seconds, both compression techniques provide a distinguishable insignia.

For objective three, with GIF89a and all three loss rates, it is not until at 15 seconds some portion of the background detail is available. The amount of background available decreases as the loss rate increases. With GIFNCa, on the other hand, it takes only 10 seconds to see some portion of the background detail. After 15 seconds almost all background detail is available.

While more serious and exhaustive empirical study is required, these initial results show some of the potential benefit of using GIFNCa over GIF89a under lossy network conditions.

4 Conclusion and Future Study

The proposed modification of any standard, particularly one as long-standing and widely accepted as GIF89a, is difficult to argue for marketing reasons regardless of any technical improvement. Initial results indicate GIFNCa's compression ratio is lower than that of GIF89a, although the degradation is minimal for the typical Ethernet MTU-size and not too costly for the typical Internet MTU size.

The gain of GIFNCa— and we extrapolate— of any network-conscious compression technique occurs when images need to be progressively displayed at the receiver as soon as possible. In military applications, seconds may be a matter of life and death. In less critical, yet still 'timely' applications such as browsing the Web, faster display will improve user perception and acceptance, allowing more successful advertising.

A primary motivation of this research is to argue that future image compression standards take into consideration whether or not the images are likely to be transmitted over the Internet and displayed in either real-time or interactive environments where progressive display efficiency is a major consideration. Network-conscious image compression focuses not simply on maximizing compression; it focuses on optimizing overall progressive display performance.

The actual gains in progressive display achieved in practice will depend on several factors.

- As *network loss* increases, the amount of disorder in IP's delivery of packets is expected to increase. This should increase the amount of information delivered and displayed earlier using a network-conscious approach. Since the loss rate increases directly with network congestion, and today's Internet congestion appears to be increasing as user demand outpaces the increases in network resources, network-conscious image compression techniques should gain in importance in the future. Furthermore, wireless networks, which are becoming more available, motivate network-conscious image compression because they suffer from high bit error rates and hand-off

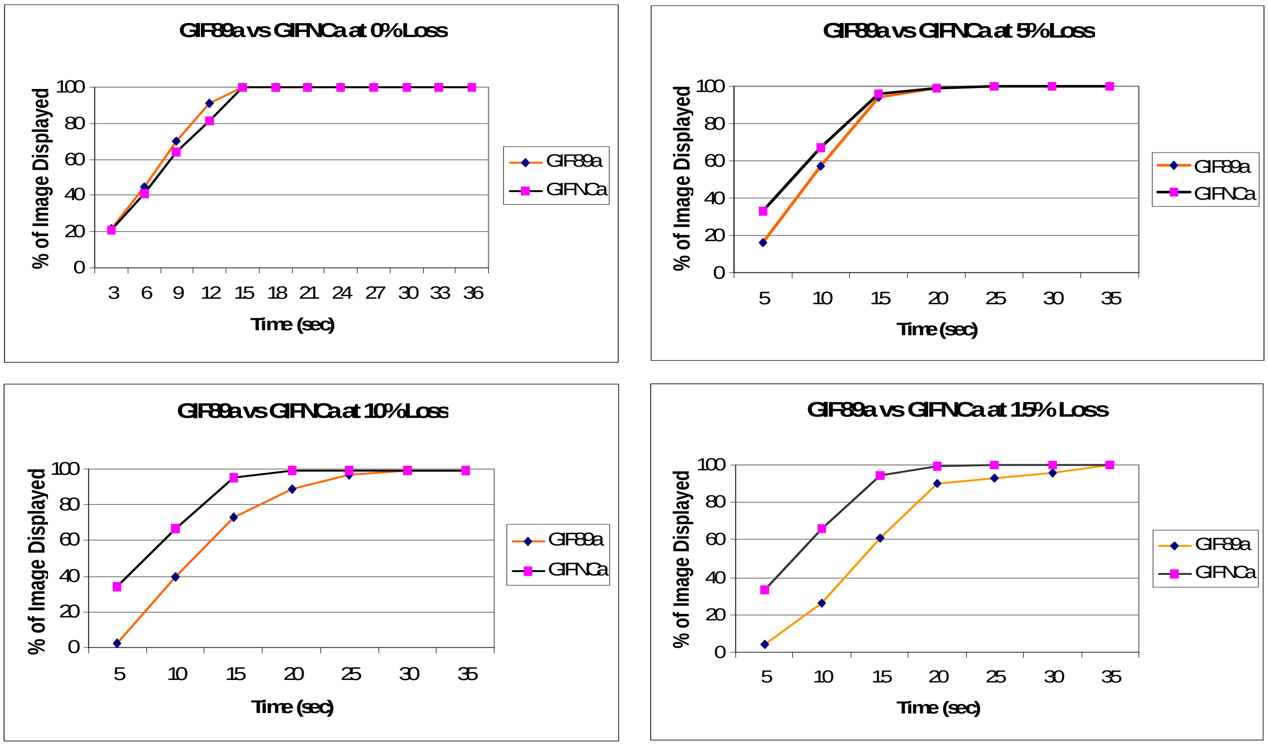

Figure 6: Comparison of GIF89a and GIFNCa at Various Loss Rates

problems resulting in even more packet losses [11].

- As *transmission delay* between end points increases, so too will the expected time between image updates thereby favoring a network-conscious approach. Transmission delay is influenced not only by the physical distance between endpoints, but also by the delays incurred within the connection path's intermediate routers performing storage and forwarding. If the present demands on the Internet continue, the end-to-end transmission delays can be expected to increase.

- At low *transmission bandwidths*, longer round trip communication slows acknowledgment feedback to the sender about lost packets. This increases the time between image updates thus enhancing network-conscious approaches' earlier delivery. While most Internet links are experiencing increases in transmission bandwidth, there remain cases where link speeds remain relatively slow (e.g., battlefield conditions using combat net radios, PPP and wireless links).

One criticism of this work may be that our results do not take into account any transport layer congestion control. We first wanted to demonstrate the primary advantage of network-conscious image compression. We are currently upgrading our experimental transport protocols, SP and XP, to include TCP-compatible congestion control. This will make them both useful extensions to the Internet's suite of transport protocols and at the same time good Internet citizens. Once these protocols are extended, we will evaluate the secondary effects of congestion control on GIFNCa vs. GIF89a.

We have been also investigating other progressive compression techniques to determine if they can be improved by being made network-conscious. In particular we are looking at wavelet encoding [10, 16], a method that does not yet have an official standard, but does have a widely accepted approach [17, 18]. We hope to demonstrate the value of network-consciousness in time to have these ideas included in any future standard.

References

[1] R. Braden. Requirements for Internet Hosts – Communication Layers. RFC 1122, October 1989.

[2] P. Conrad, P. Amer, E. Golden, S. Iren, R. Marasli, and A. Caro. Transport QoS over Unreliable Networks: No Guarantees, No Free Lunch! In Nahrstedt Campbell, ed, *Building QoS into Distributed Systems*. Chapman and Hall, 1998. `www.eecis.udel.edu/~amer/PEL/poc/postscript/iwqos97.ps`.

[3] P. Conrad, P. Amer, M. Taube, G. Sezen, S. Iren, and A. Caro. Testing Environment for Innovative Transport Protocols. In *MILCOM '98*, Bedford, MA, October 1998. (To appear).

[4] P. Conrad, E. Golden, P. Amer, and R. Marasli. A Multimedia Document Retrieval System Using Partially-Ordered/Partially-Reliable Transport Service. In *Multimedia Computing and Networking 1996*, San Jose, CA, January 1996. `www.eecis.udel.edu/~amer/PEL/poc/postscript/mmcn96full.ps`.

[5] D. Clark and D. Tennenhouse. Architectural Considerations for a New Generation of Protocols. In *ACM SIGCOMM '90*, 200–208, Philadelphia, PA, September 1990.

[6] W. Dabbous and C. Diot. High Performance Protocol Architecture. In *IFIP Performance of Computer Networks Conference (PCN '95)*, Istanbul, Turkey, October 1995. IFIP.

[7] C. Diot and F. Gagnon. Impact of Out-of-Sequence Processing on Data Transmission Performance. Tech Report Project RODEO RR-3216, INRIA - Sophia Antipolis, France, July 1997. `ftp://www.inria.fr/rodeo/diot/rr-oos.ps.gz`.

[8] Graphics Interchange Format, Version 89a. Technical report, Compuserve Incorporated, Columbus, Ohio, July 1989.

[9] E. Golden. TRUMP: Timed-Reliability Unordered Message Protocol, December 1997. MS Thesis, CIS Dept., University of Delaware.

[10] M. Hilton, B. D. Jawerth, and A. Sengupta. Compressing still and moving images with wavelets. *Multimedia Systems*, 2, 218–227, December 1994.

[11] S. Iren, P. Amer, and P. Conrad. Network-Conscious Compressed Images over Wireless Networks. In *5th International Workshop on Interactive Distributed Multimedia Systems and Telecommunication Services (IDMS'98)*, Oslo, Norway, September 1998. (To appear).

[12] S. Iren. Network-Conscious Image Compression. PhD Dissertation, CIS Dept., University of Delaware, (in progress).

[13] R. Marasli, P. Amer, and P. Conrad. Retransmission-Based Partially Reliable Services: An Analytic Model. In *IEEE INFOCOM*, San Fransisco, CA, March 1996. `www.eecis.udel.edu/~amer/PEL/poc/postscript/infocom96.ps`.

[14] J. Postel. User Datagram Protocol. RFC 768, August 1980.

[15] J. Postel. Transmission Control Protocol. RFC 793, September 1981.

[16] J. Shapiro. Embedded Image Coding Using Zerotrees of Wavelet Coefficients. *IEEE Transactions on Image Processing*, 41(12), 3445–3462, December 1993.

[17] A. Said and W. A. Pearlman. An Image Multiresolution Representation for Lossless and Lossy Image Compression. *IEEE Transactions on Image Processing*, 5, 1303–1310, September 1996.

[18] A. Said and W.A. Pearlman. A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees. 6(3), June 1996.

[19] T. A. Welch. A Technique for High-Performance Data Compression. *IEEE Computer*, 8–19, June 1984.

[20] J. Ziv and A. Lempel. Compression of Individual Sequences via Variable-Rate Coding. *IEEE Trans. on Information Theory*, 24(5), 530–536, September 1978.

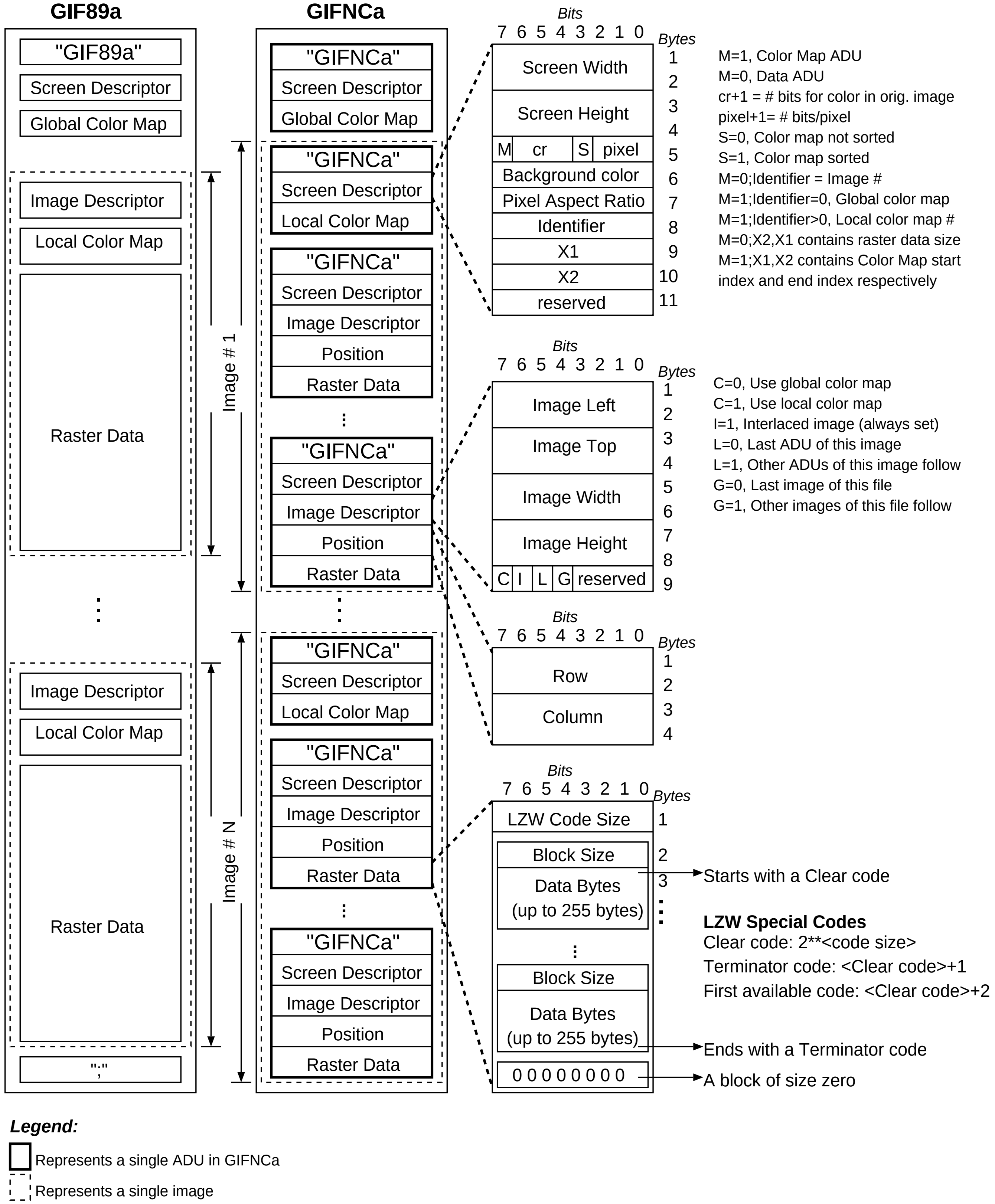

Figure 7: GIF89a vs. GIFNCa File Structure

Figure 8: Images for 5% loss

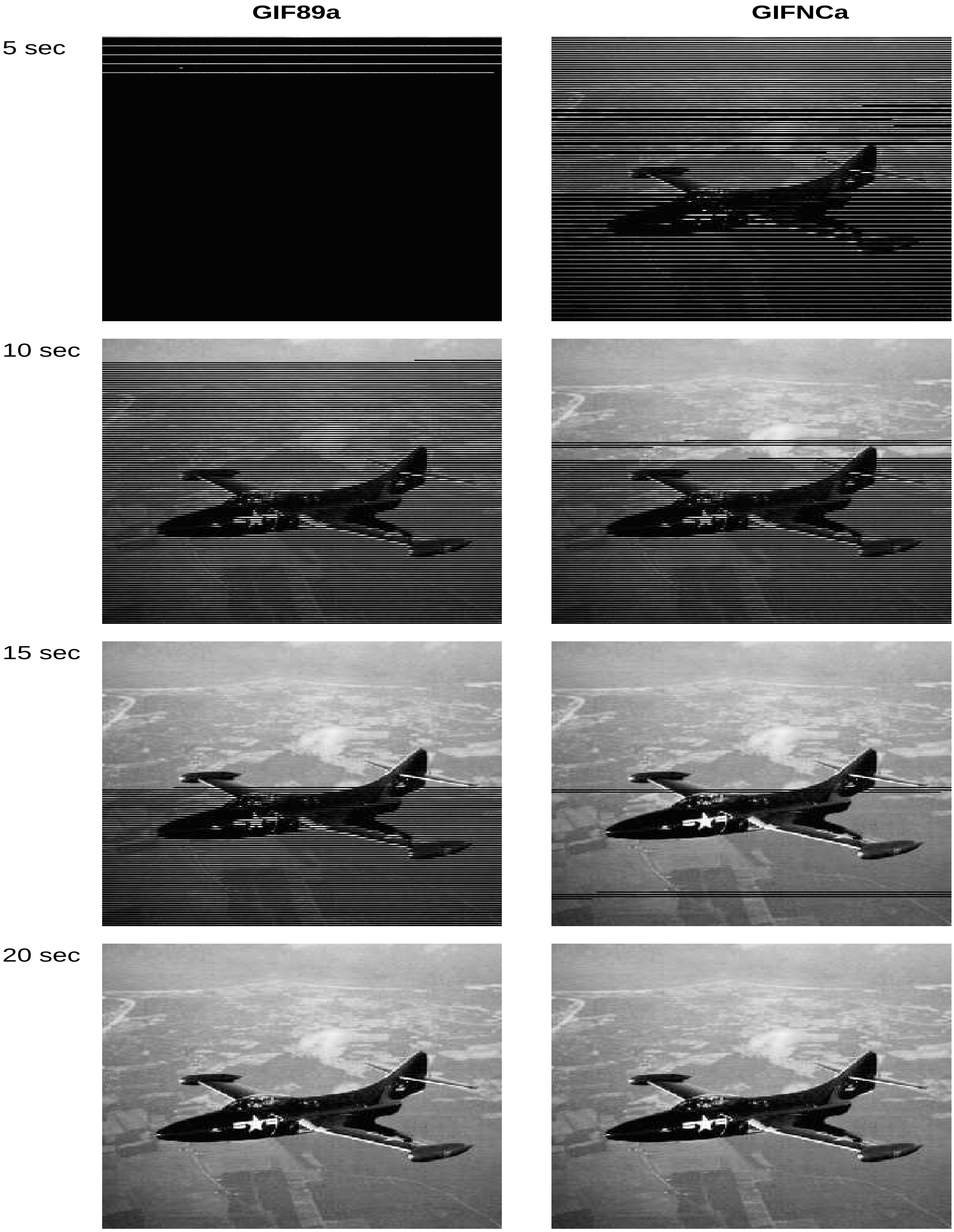

Figure 9: Images for 10% loss

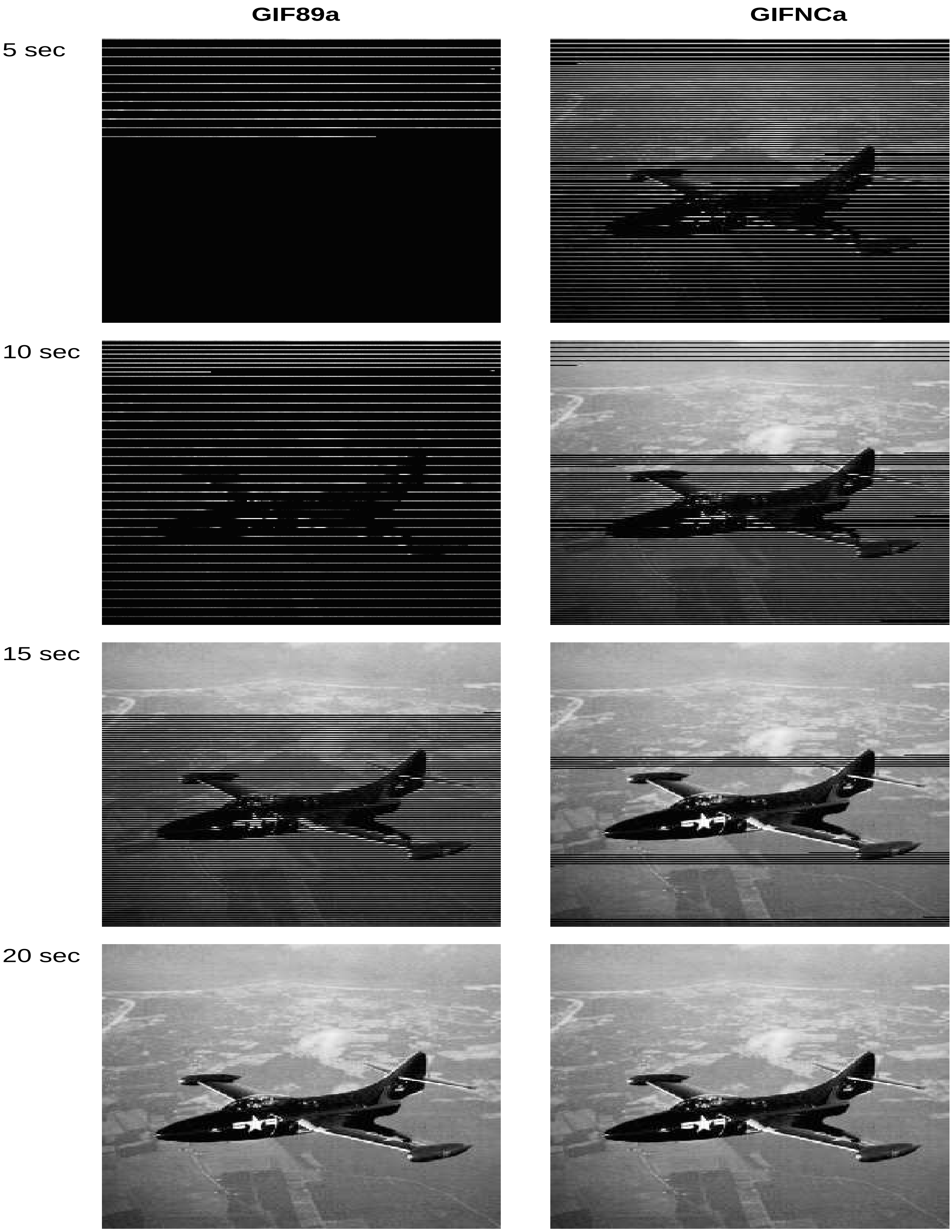

Figure 10: Images for 15% loss