

TABLE 2-continued

WORD height[1]	/* Height of image */
DWORD thrange[1]	/* Transparency range */
DWORD thrange[1]	/* Transparency range */
WORD pause [1]	/* Hundredths of seconds to pause */
BYTE packed[1]	/* Bit one is transparency flag */
WORD cid_size[1]	/* Size of combination image descriptor */
BYTE reserved[1]	/* Reserved */
} IMAGE DATAHEADER;	

[0061] The Combination Image Descriptor follows the above described image header and contains an array of vectors, which mathematically describe the Image Data 1222. Some of the Image Data 1222 may be compressed as a lossy image type while other parts of the image may be compressed a lossless image type. WORD vectorArray [cid₁₃ size]. This field size is variable and is given in the cid size field. This is an array of vectors describing lossless portions of an image. The vectors always represent a rectangular area and each one is the coordinate for its place in the image.

[0062] The n-Bit Compression Header contains six fields and is always 10 bytes in size. The first field is the size of the n-bit header. The second field contains the size of the original file. Byte order is a number 0-20, which is the order of the starting statistics for the n-bit image compression. If the fourth field BYTE adaptive [1] is turned on then the starting order adjusts during compression to the optimal level. In other words the order statistics adapt to the data in the file and compress such data to the appropriate level. WORD mask[1] contains the block size of data that will be processed before the compression ratio is checked to see if the statistics need to be flushed.

[0063] Table 3 presents the aforementioned information regarding information in the n-Bit Compression Header.

TABLE 3

typedef struct nbitcompressionheader	
{	
BYTE tsize[1]	/* Size of n-bit header */
DWORD size[1]	/* Size of original file */
BYTE order[1]	/* 0 thru 20 */
BYTE adaptive[1]	/* 1 is on, 0 is off */
WORD mask[1]	/* Block sizes */
BYTE reserved[1]	/* Reserved */
} NBITCONPRESSIONHEADER;	

[0064] Clients, represent the users of the content distribution system which may range from a simple plug to access the distribution network to a desktop version of the distribution server, with any number of variations in between. In a preferred embodiment, it may be preferable for clients to perform several functions, depending on the requirements. To perform these functions, clients may have plugins, applets, active X-components, Javascript components, HTML components and tags, client/server applications, proxies, winsock applications or services, firewall applications or servers, drivers, and/or other services. The basic functionality of a client 540 is to decompress the compressed content received from the content distribution servers and/or network. Clients may also have the capability to communicate with distribution servers through a variety of

communication protocols in order to speed up data transfer, the ability to perform client/server functions, and the ability to perform other essential or optional features. Clients may have the capability to subscribe to media multicasts from distribution server, display the media from multicasts, perform proxy functions, perform firewall functions, and/or other capabilities depending on the equipment deployed. Clients may include personal computers (PCs), handheld devices, wireless phones, etc.

[0065] FIG. 13 illustrates a diagram of the Proxy Client 1300 according to an embodiment of the invention. The Proxy Client resides on the requestor's computer; such computer can be part of a network and configured as a workstation or can be communicating with a network through an ISP. The Proxy Client 1306 is multithreaded 1308, 1310 thereby allowing several actions to occur simultaneously. The requester makes a request 1304 through a browser 1302 to the network for information, such information being made up of HTML and Images 1312. The User To Proxy Threads 1308 transfer data to the Proxy Server Threads which carry the request to the Control Server 1314 to be compressed, cached, indexed and stored in compressed or original form, or both. The control server returns compressed HTML and compressed images ("trans") 1316 to the Proxy Client 1306 which then decodes the compressed data 1318, modifies the HTTP Header 1320 and passes the decompressed data back to the browser 1322, 1324, 1326. Step 1320 converts content type such as image/transfinity, which is inbound from the control server to image/gif or image/x-bitmap, PNG, or JPEG, and changes content length to uncompressed length. In some instances undecoded information is passed back to the browser 1328.

[0066] FIG. 14 is an overview diagram of the Plugin 1400 according to an embodiment of the invention. The Plugin resides on the requestor's computer; such computer can be part of a network and configured as a workstation or can be communicating with a network through an ISP. The Plugin 1402 is multithreaded 1404, thereby allowing several actions to occur simultaneously. The requester makes a request 1406 to a network for information, such information comprising, for example, HTML and Images. The request is processed compressed and placed in a cache 1408. The control server returns compressed HTML and compressed images ("trans") 1410 to the Plugin 1402 which then decodes the compressed data 1412, modifies the HTTP Header 1414 and passes the decompressed data back to the browser 1416. The actions of the plugin are more fully described in FIG. 15.

[0067] FIG. 15 is a diagram of the internal workflow of the plugin according to an embodiment of the invention. NPN_New Create Instance and Initialize 1502 begins operating when an embed tag whose type (image/Transfinity-trans) is received from the Control Server. The plugin extracts the SRC designator whose contents are "filename.gif.trans". The original extension of the image is left as part of the SRC designator in order to prevent confusion on the part of the user. For example a given URL may have both a GIF and a JPEG of the same picture. Thus by leaving the original designation as part of the ".trans" image the user is able to differentiate between the original images. The Height and Width characteristic is also a part of the embedded information received by NPN_New Create Instance and Initialize. If such height and width information is not pro-