

```

31:  GetArg(&w, 160);
32:  GetArg(&h, 120);
33:  GetArg(&cw, 1);
34:
35:  displayAspect = (float)w / (float)h;
36:  photoAspect = (float)(img->width) / (float)(img->height);
37:
38:  if ((displayAspect > 1.0f && photoAspect < 1.0f) ||
39:      (displayAspect < 1.0f && photoAspect > 1.0f))
40:  {
41:      if (cw)
42:          IMG_RotateRight(img);
43:      else
44:          IMG_RotateLeft(img);
45:  }
46: }
47:

```

The above AutoRotateOp function automatically rotates an image to better fit the available display of a particular device. As shown on line 26 above, the function receives a pointer to an image as a parameter. On lines 28 to 36, the display characteristics of the device are obtained. The condition on lines 38 to 39 evaluates whether or not the image should be presented in portrait orientation (i.e., to display the image vertically) or landscape orientation (i.e., to display the image horizontally) to better fit the device display. Depending on the outcome of this evaluation, a call is made to IMG RotateRight or IMG RotateLeft as shown on lines 41 to 44 and the image is rotated either clockwise or counterclockwise to fit the display of a particular device. Source code listings providing further details on the IMG RotateRight and IMG RotateLeft functions are attached hereto as Appendix A.

The MTM uses cached versions of the original media objects whenever possible. The MTM caches source media objects in the backside cache to reduce the time required to read the source media and to reduce Internet bandwidth consumption by the media delivery system. This backside caching can be performed by the MTM or by an intermediate reverse proxy cache. The source objects are cached according to the directives specified in their HTTP cache-control headers. In the currently preferred embodiment, an Apache HTTP server configured as a reverse proxy cache is deployed "between" the Internet and the MTM module, so any requests for source multimedia documents are first compared to a local disk cache. The Apache reverse proxy cache module decouples the caching task from the media transformation task for better scalability and maintainability.

Appended herewith is Computer Program Listing Appendix A containing source listings, in the C/C++ programming language, providing further description of the present invention. A suitable environment for creating and building C/C++ programs is available from a variety of vendors, including Borland® C++ Builder available from Borland Software Corporation of Scotts Valley, California, and Microsoft® Visual C++ available from Microsoft Corporation of Redmond, WA.

While the invention is described in some detail with specific reference to a single-preferred embodiment and certain alternatives, there is no intent to limit the invention to that particular embodiment or those specific alternatives. For instance, those skilled in the art will appreciate that modifications may be made to the preferred embodiment without departing from the teachings of the present invention.

COMPUTER PROGRAM LISTING APPENDIX A

IMGXFORM

```
#include "vimage.h"
#include "imglib.h"

IMGLIB_API void IMG_FlipH(IMG_image *img)
{
    if (img->pixelBytes != 4)
    {
        DEBUG_IMG_FORMAT;
        return;
    }

    int32 x, y, w;
    uint8 *rowPtr;
    uint32 *pixPtr, *pixPtr2, tempPix;

    w = img->width >> 1;
    rowPtr = img->baseAddr;
    for (y = 0; y < img->height; y++)
    {
        pixPtr = pixPtr2 = (uint32 *)rowPtr;
        pixPtr2 += img->width - 1;
        for (x = 0; x < w; x++)
        {
            tempPix = *pixPtr;
            *pixPtr++ = *pixPtr2;
            *pixPtr2-- = tempPix;
        }
        rowPtr += img->rowBytes;
    }
}

IMGLIB_API void IMG_FlipV(IMG_image *img)
{
    img->baseAddr += img->rowBytes * (img->height - 1);
    img->rowBytes = -img->rowBytes;
}

IMGLIB_API void IMG_RotateLeft(IMG_image *img)
{
    if (img->pixelBytes != 4)
    {
        DEBUG_IMG_FORMAT;
        return;
    }

    IMG_image tempImg;

    if (!IMG_DuplicateImage(img, &tempImg))
        return;

    int32 x, y;
    uint8 *rowPtr, *srcRow, *srcPix;
    uint32 *pixPtr;

    if (img->rowBytes < 0)
    {
        img->baseAddr += (img->height - 1) * img->rowBytes;
        img->rowBytes = -img->rowBytes;
    }
    img->width = tempImg.height;
    img->height = tempImg.width;
    img->rowBytes = img->pixelBytes * img->width;
    rowPtr = img->baseAddr;
    srcRow = tempImg.baseAddr + tempImg.pixelBytes * (tempImg.width -
1);
    for (y = 0; y < img->height; y++)
    {
```

```

        pixPtr = (uint32 *)rowPtr;
        srcPix = srcRow;
        for (x = 0; x < img->width; x++)
        {
            *pixPtr = *((uint32 *)srcPix);
            pixPtr++;
            srcPix += tempImg.rowBytes;
        }
        rowPtr += img->rowBytes;
        srcRow -= tempImg.pixelBytes;
    }
    IMG_FreeImage(&tempImg);
}

IMGLIB_API void IMG_RotateRight(IMG_image *img)
{
    if (img->pixelBytes != 4)
    {
        DEBUG_IMG_FORMAT;
        return;
    }

    IMG_image tempImg;
    if (!IMG_DuplicateImage(img, &tempImg))
        return;

    int32 x, y;
    uint8 *rowPtr, *srcRow, *srcPix;
    uint32 *pixPtr;

    if (img->rowBytes < 0)
    {
        img->baseAddr += (img->height - 1) * img->rowBytes;
        img->rowBytes = -img->rowBytes;
    }
    img->width = tempImg.height;
    img->height = tempImg.width;
    img->rowBytes = img->pixelBytes * img->width;
    rowPtr = img->baseAddr;
    srcRow = tempImg.baseAddr + tempImg.rowBytes * (tempImg.height - 1);
    for (y = 0; y < img->height; y++)
    {
        pixPtr = (uint32 *)rowPtr;
        srcPix = srcRow;
        for (x = 0; x < img->width; x++)
        {
            *pixPtr = *((uint32 *)srcPix);
            pixPtr++;
            srcPix -= tempImg.rowBytes;
        }
        rowPtr += img->rowBytes;
        srcRow += tempImg.pixelBytes;
    }
    IMG_FreeImage(&tempImg);
}

IMGLIB_API void IMG_Rotate180(IMG_image *img)
{
    IMG_FlipV(img);
    IMG_FlipH(img);
}

IMGLIB_API void IMG_Resize(IMG_image *img, int32 newWidth, int32
newHeight, bool highQuality)
{
    if (img->pixelBytes != 4)
    {
        DEBUG_IMG_FORMAT;
        return;
    }

```



```

        if (img->width == newWidth && img->height == newHeight)
            return;

        IMG_image tempImg;

        if (!IMG_AllocateImage(&tempImg, newWidth, newHeight, img-
>imgFormat))
            return;

        //IMG_StretchBlit(img, &tempImg, 0, 0, newWidth, newHeight, true);
        if (highQuality)
        {
            VImage vImg;
            vImg.Import(img);
            vImg.Scale(newWidth, newHeight, CImageServer::CUBIC);
            vImg.Export(&tempImg);
        }
        else
            IMG_StretchBlit(img, &tempImg, 0, 0, newWidth, newHeight,
false);

        IMG_CopyTags(img, &tempImg);
        IMG_FreeImage(img);
        *img = tempImg;
    }

IMGLIB_API void IMG_ClampResize(IMG_image *img, int32 maxWidth, int32
maxHeight, bool highQuality)
{
    float f, aspect, newAspect;
    int32 width, height;

    aspect = (float)(img->width) / (float)(img->height);
    newAspect = (float)maxWidth / (float)maxHeight;
    if (aspect > newAspect)
    {
        width = maxWidth;
        f = (float)width / aspect;
        height = ROUND(f);
        height = CLAMP(height, 1, maxHeight);
    }
    else
    {
        height = maxHeight;
        f = (float)height * aspect;
        width = ROUND(f);
        width = CLAMP(width, 1, maxWidth);
    }
    IMG_Resize(img, width, height, highQuality);
}

```

Apache Module

```

/**
 * This function creates a device capabilities object
 * and constructs the URL that is passed to the
 * Media Transcoding Module.
 *
 * @param r
 * @return
 */
static int uts_rewrite_uri(request_rec *r) {
    int status;

    // skip if already a proxy
    if (r->proxyreq != NOT_PROXY) {
        return OK;
    }
}

```

```

    // skip all but GET requests
    if (strcasecmp("GET", r->method) != 0) {
        return OK;
    }

    // get config file
    uts_dir_config *cfg = (uts_dir_config *) ap_get_module_config(r-
>per_dir_config, &uts_module);
    uts_server_config *scfg = (uts_server_config *)
ap_get_module_config(r->server->module_config, &uts_module);

    // (subrequest for content type)
    string contentType = "image/jpeg";

    /**
     * first, parse the full incoming URI. We're going to treat the
path+args
     * of this URI as the full URI for the proxy request
     */
    uri_components uc;
    status = ap_parse_uri_components(r->pool, r->uri, &uc);
    if (status != HTTP_OK) {
        ap_log_error(APLOG_MARK, APLOG_ERR | APLOG_NOERRNO, r->server,
"Bad input URI: %s", r->uri);
        return DECLINED;
    }

    char *path = ap_pstrdup(r->pool, uc.path);

    // pass in headers, get device capabilities back
    DeviceIdentifier::DeviceCapabilities devcap;
    status = dev_ident->getDeviceCapabilities(&devcap, r, contentType);
    if (status != DI_SUCCESS) {
        ap_log_error(APLOG_MARK, APLOG_ERR | APLOG_NOERRNO, r->server,
"Device capabilities lookup failed");
        ap_log_error(APLOG_MARK, APLOG_ERR | APLOG_NOERRNO, r->server,
"Recording headers");
        logHeaders(r);
        return DECLINED;
    }

    ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, r->server, "%s",
devcap.toString().c_str());

    char *sub;
    // pop off initial slash
    sub = ap_getword_nc(r->pool, &path, '/'); // initial slash

    // if we have an additional subscriber ID, pop it off, too
    if (cfg->subscriber_id_on_url) {
        sub = ap_getword_nc(r->pool, &path, '/'); // subscriber ID
    }

    //
    // Build option list.
    // TODO: This should really be a function that returns a string.
    //
    char *cfg_clamp, *cfg_mimetype, *cfg_path, *cfg_quality,
*cfg_argparam, *cfg_rotate, *cfg_scale, *cfg_filesize;
    char blank = '\0';

    // Clamp or scale?
    if (!devcap.scale_to_width)
    {
        cfg_clamp = ap_psprintf(r->pool, "%sclampsize=%d&",
            (strchr(cfg->media_transcoder_uri, '?') != NULL ? "&" : "?"),
            (devcap.display_height > devcap.display_width ?
devcap.display_height : devcap.display_width));
        cfg_scale = ap_psprintf(r->pool, "%c", blank);
    } else {
        cfg_clamp = ap_psprintf(r->pool, "%s",

```

```

        (strchr (cfg->media_transcoder_uri, '?') != NULL ? "&" : "?"));
        cfg_scale = ap_psprintf (r->pool, "limitsize=%i,0&",
devcap.display_width) ;
    }

    // Path
    if (path[strlen(path)-2] == '.')
    {
        switch (path[strlen(path)-1])
        {
            case 'b':
                cfg_path = ap_psprintf (r->pool, "in=\"http://%smp\"&", path) ;
                break ;
            case 'g':
                cfg_path = ap_psprintf (r->pool, "in=\"http://%sif\"&", path) ;
                break ;
            case 'j':
                cfg_path = ap_psprintf (r->pool, "in=\"http://%spg\"&", path) ;
                break ;
            case 'p':
                cfg_path = ap_psprintf (r->pool, "in=\"http://%sng\"&", path) ;
                break ;
            case 't':
                cfg_path = ap_psprintf (r->pool, "in=\"http://%sif\"&", path) ;
                break ;
        }
    }
    else {
        cfg_path = ap_psprintf (r->pool, "in=\"http://%s\"&", path) ;
    }

    // Mime Type
    cfg_mimetype = ap_psprintf (r->pool, "outformat=%s&",
devcap.mime_type.c_str()) ;

    // Quality
    if (devcap.image_quality != -1)
        cfg_quality = ap_psprintf (r->pool, "outquality=%i&",
devcap.image_quality) ;
    else
        cfg_quality = ap_psprintf (r->pool, "outquality=%i&",
devcap.color_depth) ;

    // Rotate?
    if (devcap.rotate_to_fit)
        cfg_rotate = ap_psprintf (r->pool, "autorotate=%i,%i,0&",
devcap.display_width,
devcap.display_height) ;
    else
        cfg_rotate = ap_psprintf (r->pool, "%c", blank) ;

    // Arguments
    if (r->args)
        cfg_argparam = ap_psprintf (r->pool, "%s&", r->args) ;
    else
        cfg_argparam = ap_psprintf (r->pool, "%c", blank) ;

    // Filesize
    if (devcap.maximum_file_size)
        cfg_filesize = ap_psprintf (r->pool, "filesize=%i&",
devcap.maximum_file_size) ;
    else
        cfg_filesize = ap_psprintf (r->pool, "%c", blank) ;

    //
    // End option list build
    //

    // create URL for lps
    //
    [1]URL[2]Clamp[3]path[4]format[5]quality[6]rotate[7]argparam[8]scale[9]fil
esize

```

```

char *url = ap_psprintf (r->pool, "%s%s%s%s%s%s%s%s",
    cfg->media_transcoder_uri,
    cfg_clamp,
    cfg_path,
    cfg_mimetype,
    cfg_quality,
    cfg_rotate,
    cfg_argparam,
    cfg_scale,
    cfg_filesize) ;

ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, r->server,
    "proxying: %s", url);

// double-check that this is a valid URL
status = ap_parse_uri_components(r->pool, url, &uc);
if (status != HTTP_OK) {
    ap_log_error(APLOG_MARK, APLOG_ERR | APLOG_NOERRNO, r->server,
        "Bad proxy URI: %s", url);
    return DECLINED;
}

// internal redirect won't work for remote servers, so
// set this up as a proxy
/* example of working request member variables
r->proxyreq      = PROXY_PASS;
r->uri           = "/lsp";
r->filename      = "proxy:http://pegli-
nt4.office.lightsurf.com:10104/lsp";
r->args          =
"clampsize=100&in=http://www.lightsurf.com/images/misc/pk_sunset.jpg";
r->handler       = "proxy-server";
*/

r->proxyreq      = PROXY_PASS;
r->uri           = ap_pstrdup(r->pool, uc.path);
r->filename      = ap_pstrcat(r->pool, "proxy:", uc.scheme, "://",
uc.hostinfo, uc.path, NULL);
r->args          = ap_pstrdup(r->pool, uc.query);
r->handler       = "proxy-server";

return DECLINED;
}

```

DeviceIdentifier

```

/**
 * DeviceIdentifier.cpp
 * implementation of DeviceIdentifier and DeviceCapabilities classes
 * 4/23/2001 pae
 */

#include <string>
#include <sstream.h>
#include <stdlib.h>

#include "httpd.h"
#include "http_log.h"

#include "Regex.hpp"
#include "DeviceIdentifier.hpp"

static Regex regex;

DeviceIdentifier::DeviceIdentifier(const char * filename) {
    try {
        config = new XMLConfigFile(filename);
    }
}

```

```

    }
    catch (...) {
    }
}

DeviceIdentifier::DeviceIdentifier() {
    config = new XMLConfigFile("/lsurf/uts/conf/devices.xml");
}

DeviceIdentifier::~~DeviceIdentifier() {
    if (config != NULL) {
        delete config;
    }
}

/**
 * Retrieve a named capability value from the configuration
 * file. This function will first find the <capability>
 * node identified by <code>capabilityName</code>, then
 * determine if any <header> tags apply. The
 * value is returned as a string, which can then be
 * converted to the appropriate type.
 *
 * @param r
 * @param baseXPath
 * @param capabilityName
 * @return
 */
string DeviceIdentifier::getCapability(request_rec *r, string baseXPath,
string capabilityName) {
    /**
     * The device capabilities data storage is implemented as an XML file.
     * A member variable named "config" allows us to make XPath queries
     * against the XML file.
     */
    string cquery =
baseXPath +
string("/capability[@name=\"" +
capabilityName +
string("\"]");

    string value;

    // first, look through headers (if any)
    XMLConfigFile::StringVectorType headernames = config->query(cquery +
string("/header[@name=\"" + headernames[i] +
string("\"]/@pattern"));
    for (unsigned int i=0; i < headernames.size(); i++) {
        const char *headerval = ap_table_get(r->headers_in,
headernames[i].c_str());
        if (headerval != NULL) {
            // found a matching header, so grab the pattern and run the
            regex
            XMLConfigFile::StringVectorType patterns = config-
>query(cquery + string("/header[@name=\"" + headernames[i] +
string("\"]/@pattern"));
            if (regex.group(patterns[0], string(headerval), &value)) {
                // yes, there is a match between the <header pattern=">
                and the header value
                if (value.size() > 0) {
                    // value now contains the matched substring
                    break;
                } else {
                    // there was a pattern match, but no parenthesized
                    substring, so use the value of the "default" attribute
                    XMLConfigFile::StringVectorType defaults = config-
>query(cquery + string("/header[@name=\"" + headernames[i] +
string("\"]/@default"));
                    value = defaults[0];
                }
            }
        }
    }
}

```

```

    }
}

// if we didn't get anything from the headers, use the default value
XMLConfigFile::StringVectorType defaults = config->query(cquery +
string("/@default"));
if (defaults.size() > 0) {
    value = defaults[0];
}

// TODO: if there isn't a default, die a horrible death
// this may not be necessary if we publish a schema for the config
file

return value;
}

```

```

/**
 * Convert the result of <code>getCapability()</code>
 * to an integer.
 *
 * @param r
 * @param baseXPath
 * @param capabilityName
 * @return
 */
int DeviceIdentifier::getCapabilityInt(request_rec *r, string baseXPath,
string capabilityName) {
    string value = getCapability(r, baseXPath, capabilityName);
    if (value.size() > 0) {
        return atoi(value.c_str());
    } else {
        return -1;
    }
}

```

```

/**
 * Convert the result of <code>getCapability()</code>
 * to a boolean.
 *
 * @param r
 * @param baseXPath
 * @param capabilityName
 * @return
 */
int DeviceIdentifier::getCapabilityBool(request_rec *r, string baseXPath,
string capabilityName) {
    string value = getCapability(r, baseXPath, capabilityName);
    return(!value.compare("true") || !value.compare("1")); // anything
but true or 1 is considered false
}

```

```

/**
 * Fill in a DeviceCapabilites structure based on the
 * HTTP request headers and the content type of the
 * requested document. The main job of this function
 * is to find the correct node in the config file for
 * the given User-Agent header, then call
 * <code>getCapabilityStr()</code>, <code>getCapabilityInt()</code>,
 * and <code>getCapabilityBool()</code> to fill in
 * the device capabilities.
 *
 * @param devcap
 * @param r

```

```

    * @param contentType
    */
    int
    DeviceIdentifier::getDeviceCapabilities(DeviceIdentifier::DeviceCapabiliti
    es *devcap, request_rec *r, string contentType) {

        if (devcap == NULL) {
            // sorry, no can do
            ap_log_error(APLOG_MARK, APLOG_ERR, r->server, "Cannot call
            getDeviceCapabilities() with a null DeviceCapabilities parameter");
            return DI_ERROR;
        }

        XMLConfigFile::StringVectorType xpath_results;
        string base_xpath_query;

        // find the correct user-agent node
        const char *ua_header = ap_table_get(r->headers_in, "User-Agent");

        xpath_results = config->query("//user-agent/@pattern");
        int found = FALSE;
        for (unsigned int i=0; i < xpath_results.size(); i++) {
            if (regex.match(xpath_results[i], string(ua_header))) {
                base_xpath_query.append("//user-agent[@pattern=\"");
                base_xpath_query.append(xpath_results[i]);
                base_xpath_query.append("\"]");
                found = TRUE;
                break;
            } else if (regex.last_error.size() > 0) {
                // this will complain a lot if there are regex errors
                ap_log_error(APLOG_MARK, APLOG_ERR, r->server, "Regular
                expression error: %s", regex.last_error.c_str());
                ap_log_error(APLOG_MARK, APLOG_DEBUG, r->server, "Regex: %s",
                xpath_results[i].c_str());
            }
        }
        if (!found) {
            ap_log_error(APLOG_MARK, APLOG_DEBUG, r->server, "Could not find
            configuration entry for User-Agent \"%s\"", ua_header);
            // TODO: log or send email
            return DI_NOTFOUND;
        }

        // within that node, find the correct mime-type node
        xpath_results = config->query(base_xpath_query + string("/mime-
        type/@pattern"));

        found = FALSE;
        for (unsigned int i=0; i < xpath_results.size(); i++) {
            if (regex.match(xpath_results[i], contentType)) {
                base_xpath_query.append("/mime-type[@pattern=\"");
                base_xpath_query.append(xpath_results[i]);
                base_xpath_query.append("\"]");
                found = TRUE;
                break;
            } else if (regex.last_error.size() > 0) {
                // again, here's another big complainer
                ap_log_error(APLOG_MARK, APLOG_ERR, r->server, "Regular
                expression error: %s", regex.last_error.c_str());
                ap_log_error(APLOG_MARK, APLOG_DEBUG, r->server, "Regex: %s",
                xpath_results[i].c_str());
            }
        }
        if (!found) {
            ap_log_error(APLOG_MARK, APLOG_DEBUG, r->server, "Could not find
            <mime-type> entry in config file for document MIME type \"%s\"",
            contentType.c_str());
            // TODO: log or send email
            return DI_NOTFOUND;
        }
    }
}

```

```

    ap_log_error(APLOG_MARK, APLOG_DEBUG, r->server, "Successfully
identified device, User-Agent: \"%s\"", ua_header);

    // fill in the DeviceCapabilities structure
    // note defaults are: string="", int=-1, bool=false
    devcap->mime_type = getCapability(r, base_xpath_query,
"output-mime-type");
    devcap->display_width = getCapabilityInt(r, base_xpath_query,
"display-width");
    devcap->display_height = getCapabilityInt(r, base_xpath_query,
"display-height");
    devcap->color_depth = getCapabilityInt(r, base_xpath_query,
"color-depth");
    devcap->image_quality = getCapabilityInt(r, base_xpath_query,
"image-quality");
    devcap->color_capable = getCapabilityBool(r, base_xpath_query,
"color-capable");
    devcap->rotate_to_fit = getCapabilityBool(r, base_xpath_query,
"rotate-to-fit");
    devcap->scale_to_width = getCapabilityBool(r, base_xpath_query,
"scale-to-width");
    devcap->palette = getCapability(r, base_xpath_query,
"palette");
    devcap->maximum_file_size = getCapabilityInt(r, base_xpath_query,
"maximum-size");
    devcap->video_frame_rate = getCapabilityInt(r, base_xpath_query,
"video-frame-rate");
    devcap->audio_encoding_rate = getCapabilityInt(r, base_xpath_query,
"audio-encoding-rate");
    devcap->audio_channels = getCapabilityInt(r, base_xpath_query,
"audio-channels");

    return DI_SUCCESS;
};

/*****
*****/

// utility functions to print out the DeviceCapabilities object
void DeviceIdentifier::DeviceCapabilities::print(ostream *os) {
    *os << this->toString();
};

string DeviceIdentifier::DeviceCapabilities::toString() {
    std::ostringstream buffer;
    buffer << "DeviceCapabilites {"
    << " mime-type: " << mime_type
    << "; display width: " << display_width
    << "; display height: " << display_height
    << "; color depth: " << color_depth
    << "; color capable? " << (color_capable ? "yes" : "no")
    << "; rotate to fit? " << (rotate_to_fit ? "yes" : "no")
    << "; video frame rate: " << video_frame_rate
    << "; audio encoding rate: " << audio_encoding_rate
    << "; audio channels: " << audio_channels
    << " }"
    << "\0";
    return buffer.str();
};

ostream &operator<<(ostream &os, DeviceIdentifier::DeviceCapabilities
&devcap) {
    devcap.print(&os);
    return os;
};

```


WHAT IS CLAIMED IS:

1. In an online system, a method for determining the capabilities of client devices and supplying media content in a format suitable for such devices, the method comprising:
receiving a request to provide a target device with a copy of a particular media object;
determining capabilities of the target device;
based on the capabilities of the target device, determining a format that is desired for providing the target device with a copy of the media object;
translating the particular media object into a copy having said determined format; and
providing the target device with the copy having said determined format.
2. The method of claim 1, further comprising
storing the copy having said determined format in a cache memory.
3. The method of claim 2, further comprising:
receiving from a target device a subsequent request for the particular object in the determined format; and
providing the target device with the copy stored in said cache memory.
4. The method of claim 1, further comprising:
obtaining a copy of said particular media object from a connected server for translation of said media object.
5. The method of claim 4, further comprising:
storing in cache memory a cached copy of said media object received from said connected server; and
in response to subsequent requests for translation of said media object, using the copy of said media object stored in cache memory.
6. The method of claim 1, wherein the capabilities of the target device include screen resolution.

7. The method of claim 1, wherein the capabilities of the target device include screen size.
8. The method of claim 1, wherein the capabilities of the target device include color support.
9. The method of claim 1, wherein the capabilities of the target device include bit rate.
10. The method of claim 1, wherein the capabilities of the target device include currently-available communication medium that the target device employs to transmit its request.
11. The method of claim 10, wherein currently-available communication medium comprises wireless communication.
12. The method of claim 10, wherein currently-available communication medium comprises wireline communication.
13. The method of claim 1, wherein said step of determining capabilities of the target device includes examining the request submitted by the device
14. The method of claim 1, wherein said step of determining capabilities of the target device includes examining the HTTP header submitted by the device.
15. The method of claim 14, wherein examining the HTTP header submitted by the device includes examining the HTTP User-Agent header.
16. The method of claim 1, wherein said step of determining capabilities of the target device includes querying the device for its capabilities.

17. The method of claim 1, wherein said step of determining capabilities of the target device includes determining capabilities from a knowledgebase, based on a device class for the target device.

18. The method of claim 17, further comprising:
recording a log record of target devices that are not recognized to enable the capabilities of said devices to be added to the knowledgebase.

19. The method of claim 18, further comprising:
automatically issuing notifications regarding said target devices that are not recognized.

20. The method of claim 1, wherein said step of determining a format that is desired includes determining an appropriate resolution for rendering the particular image at the target device.

21. The method of claim 1, wherein said step of determining a format that is desired includes determining an appropriate color space for rendering a particular image at the target device.

22. The method of claim 1, wherein said step of determining a format that is desired includes determining an appropriate image size for rendering the particular image at the target device.

23. The method of claim 1, wherein said step of determining a format that is desired includes determining whether to rotate the particular image to conform to the aspect ratio of the target device display.

24. The method of claim 1, wherein said step of determining a format that is desired includes determining the appropriate bit rate for the target device.

25. The method of claim 1, wherein said step of determining a format that is desired includes determining communication bandwidth available for transmitting a copy of the particular media object to the target device.

26. The method of claim 25, wherein the communication bandwidth available is determined, at least in part, based on the HTTP request header received from the target device.

27. The method of claim 25, wherein the communication bandwidth available is determined, at least in part, based on a device class for the target device.

28. The method of claim 1, wherein said target device includes a handheld computing device having display capability.

29. The method of claim 1, wherein said target device includes a handheld computing device having digital audio capability.

30. The method of claim 1, wherein said target device includes a cellular phone device having display capability.

31. The method of claim 1, wherein said target device includes a cellular phone device having digital audio capability.

32. The method of claim 1, wherein said target device includes a pager device having display capability.

33. The method of claim 1, wherein said target device includes a personal computer having display capability.

34. The method of claim 1, wherein said target device includes a personal computer having digital audio capability.

35. The method of claim 1, wherein said target device includes WAP (Wireless Application Protocol) support.

36. The method of claim 1, wherein said media objects include digital images.

37. The method of claim 1, wherein said digital objects include digital video.

38. The method of claim 1, wherein said digital objects include digital audio.

39. An online system for providing digital media to target devices, the system comprising:

a capabilities module for determining the capabilities of a particular target device;

a transformation module for:

automatically retrieving a copy of a particular media object; and

providing the target device with a copy of said object, said copy being automatically translated into a particular format based on the capabilities of the target device.

40. The system of claim 39, further comprising:

a cache memory for storing copies of media objects that have been translated.

41. The system of claim 40, wherein said system first attempts to satisfy the request by retrieving a copy of the particular object in the particular format from the cache memory.

42. The system of claim 39, further comprising:

a cache memory for storing copies of media objects that have been retrieved.

43. The system of claim 42, wherein said system first attempts to retrieve a copy of the particular media object from the cache memory before retrieving a copy from a remote server.

44. The system of claim 39, wherein each digital object stored by said system is identified by a unique uniform resource locator (URL).

45. The system of claim 44, wherein said unique URL is encoded with the characteristics of said digital object.

46. The system of claim 44, wherein said unique URL includes the color depth of said digital object.

47. The system of claim 44, wherein said unique URL includes the image size of said digital object.

48. The system of claim 44, wherein said unique URL includes the resolution of said digital object.

49. The system of claim 44, wherein said unique URL includes the bit rate of said digital object.

50. The system of claim 44, wherein said system stores URLs for each of the digital objects, and wherein the capabilities module is capable of determining the particular digital object that may be provided to the target device from said URLs.

51. The system of claim 39, wherein the capabilities of the target device include screen resolution.

52. The system of claim 39, wherein the capabilities of the target device include screen size.

53. The system of claim 39, wherein the capabilities of the target device include color support.

54. The system of claim 39, wherein the capabilities of the target device include bit rate.

55. The system of claim 39, wherein the capabilities of the target device include currently-available communication medium that the target device employs to transmit its request.

56. The system of claim 55, wherein currently-available communication medium comprises wireless communication.

57. The system of claim 55, wherein currently-available communication medium comprises wireline communication.

58. The system of claim 39, wherein said capabilities module includes the ability to determine the capabilities of the target device from its HTTP header.

59. The system of claim 58, wherein said capabilities module includes the ability to determine the capabilities of the target device from its HTTP User-Agent header.

60. The system of claim 39, wherein said capabilities module includes the ability to query the target device for its capabilities.

61. The system of claim 39, wherein said capabilities module includes a knowledgebase for determining the capabilities of the target device based on its device class.

62. The system of claim 61, further comprising:
a log record for recording target devices that are not recognized to enable the capabilities of said devices to be added to the knowledgebase.

63. The system of claim 39, wherein said particular format is selected based on an appropriate resolution for rendering the particular media object at the target device.

64. The system of claim 39, wherein said particular format is selected based on an appropriate color space for rendering the particular media object at the target device.

65. The system of claim 39, wherein said particular format is selected based on an appropriate image size for rendering the particular media object at the target device.

66. The system of claim 39, wherein said particular format is selected based on an appropriate bit rate for rendering the particular media object at the target device.

67. The system of claim 39, wherein said particular format is selected based on communication bandwidth available for transmitting a copy of the particular media object to the target device.

68. The system of claim 67, wherein the communication bandwidth available is determined, at least in part, based on a device class for the target device.

69. The system of claim 39, wherein said target device includes a handheld computing device having display capability.

70. The system of claim 39, wherein said target device includes a handheld device having digital audio capability.

71. The system of claim 39, wherein said target device includes a cellular phone device having display capability.

72. The system of claim 39, wherein said target device includes a cellular phone device having digital audio capability.

73. The system of claim 39, wherein said target device includes a pager device having display capability.

74. The system of claim 39, wherein said target device includes a personal computer having display capability.

75. The system of claim 39, wherein said target device includes a personal computer device having digital audio capability.

76. The system of claim 39, wherein said target device includes WAP (Wireless Application Protocol) support.

77. In an online system, a method for determining the capabilities of client devices, the method comprising:

receiving an original request from a target device in which said target device does not include information regarding its capabilities;

determining capabilities of the target device;

supplementing said original request received from said target device with information about the capabilities of said target device; and

forwarding said supplemented request to a destination specified in said original request.

78. The method of claim 77, wherein said step of determining capabilities of the target device includes examining the request submitted by the device

79. The method of claim 77, wherein said step of determining capabilities of the target device includes examining the HTTP header submitted by the device.

80. The method of claim 79, wherein examining the HTTP header submitted by the device includes examining the HTTP User-Agent header.

81. The method of claim 77, wherein said step of determining capabilities of the target device includes querying the device for its capabilities.

82. The method of claim 77, wherein said step of determining capabilities of the target device includes determining capabilities from a knowledgebase, based on a device class for the target device.

**SYSTEM AND METHODOLOGY FOR DELIVERING MEDIA TO MULTIPLE
DISPARATE CLIENT DEVICES BASED ON THEIR CAPABILITIES**

ABSTRACT OF THE DISCLOSURE

An online media delivery system incorporating combining on-the-fly media
5 reformatting with advanced client detection capabilities enabling delivery of appropriate
media content to connected client devices is described. The system receives requests from
client devices for media documents or objects, identifies the client device requesting
particular media objects from an HTTP request, determines the media output capabilities of
the client device, reformats the source media according to those capabilities and delivers the
10 reformatted media to the client device. The system provides access to media content for
multiple disparate client devices of varying hardware and software capabilities. The system
enables delivery of appropriate media content to practically any device with connectivity
capability.

15

LS0024.00.(Ribbon 11-08-01) app.doc

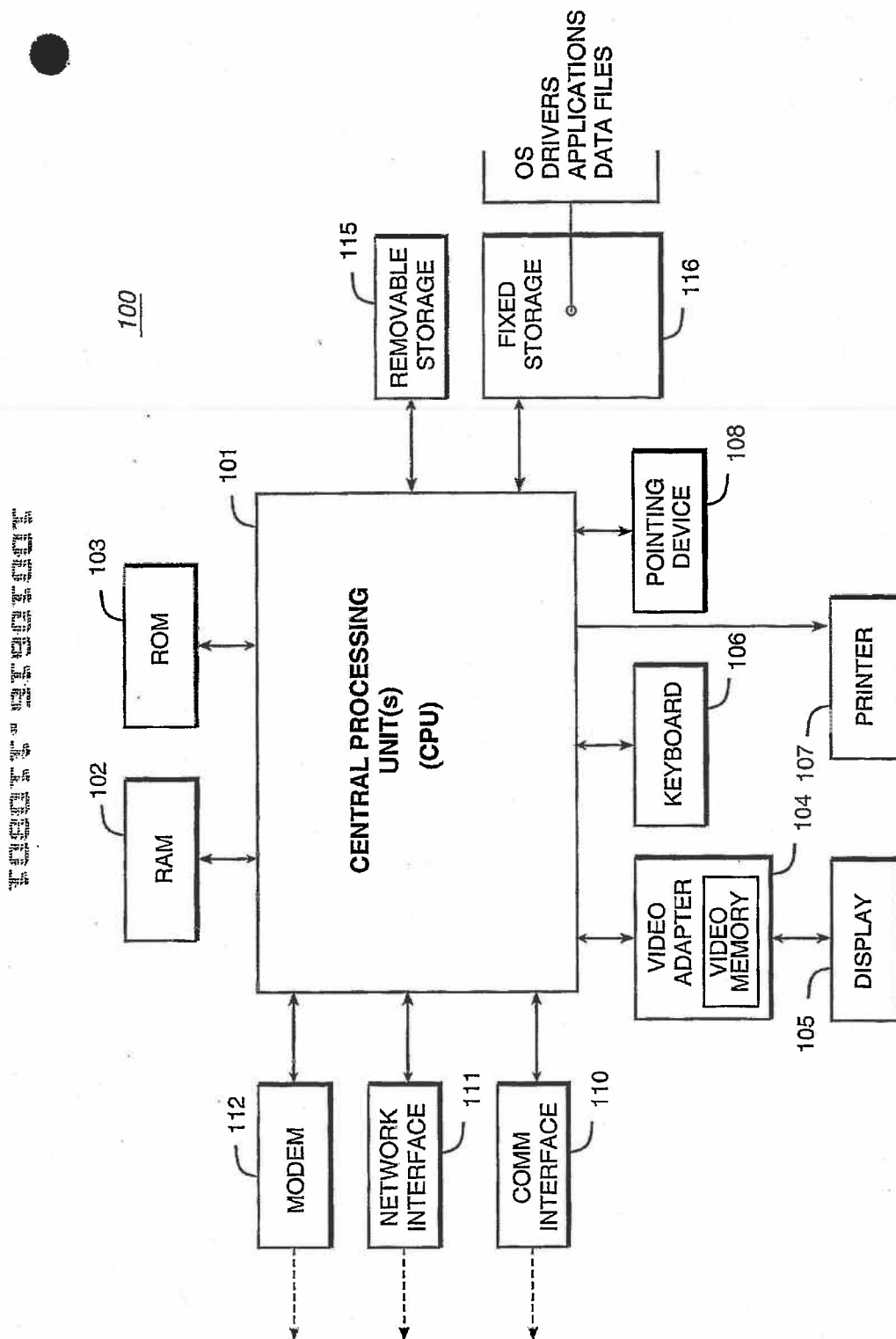


FIG. 1
(PRIOR ART)

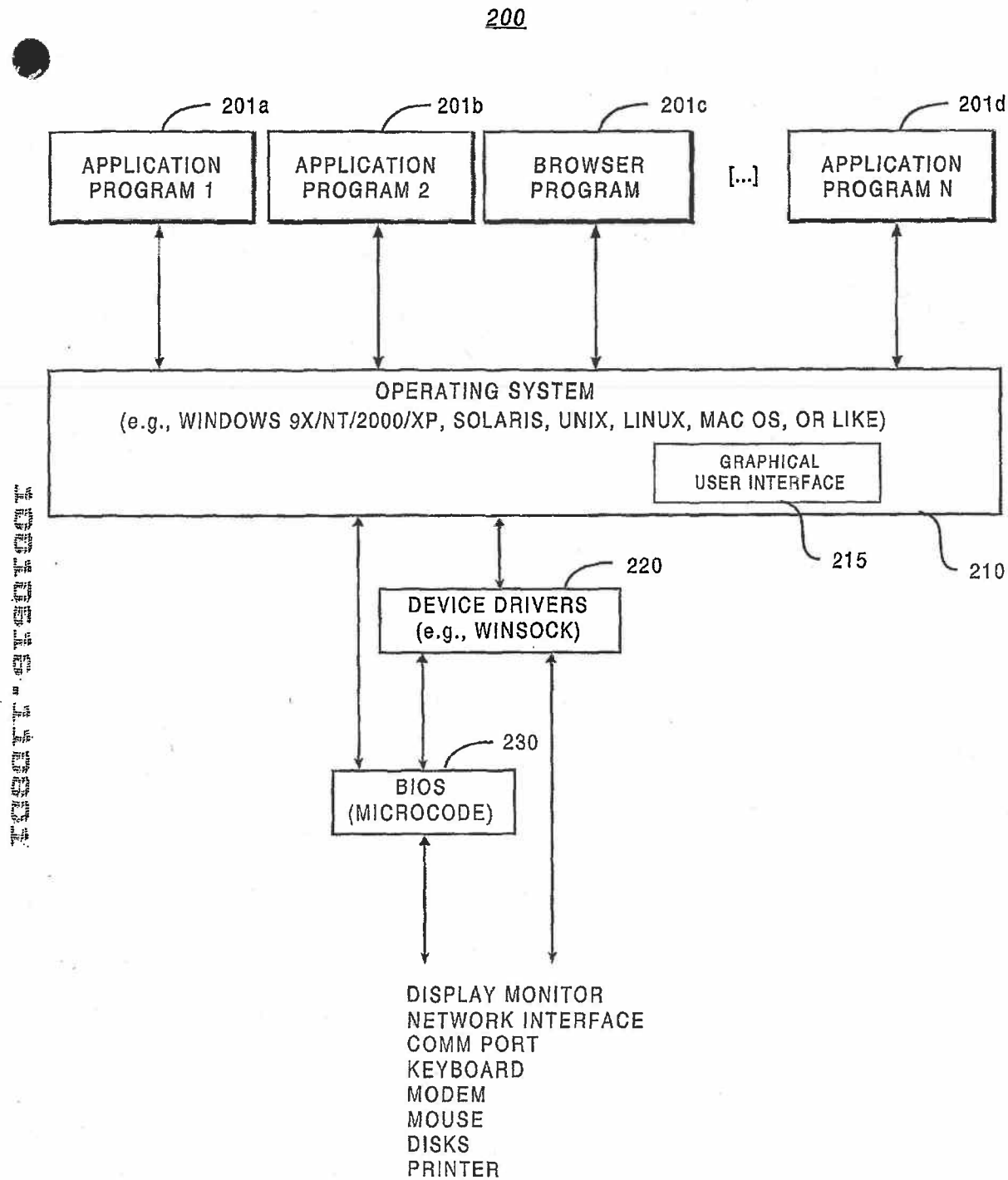


FIG. 2

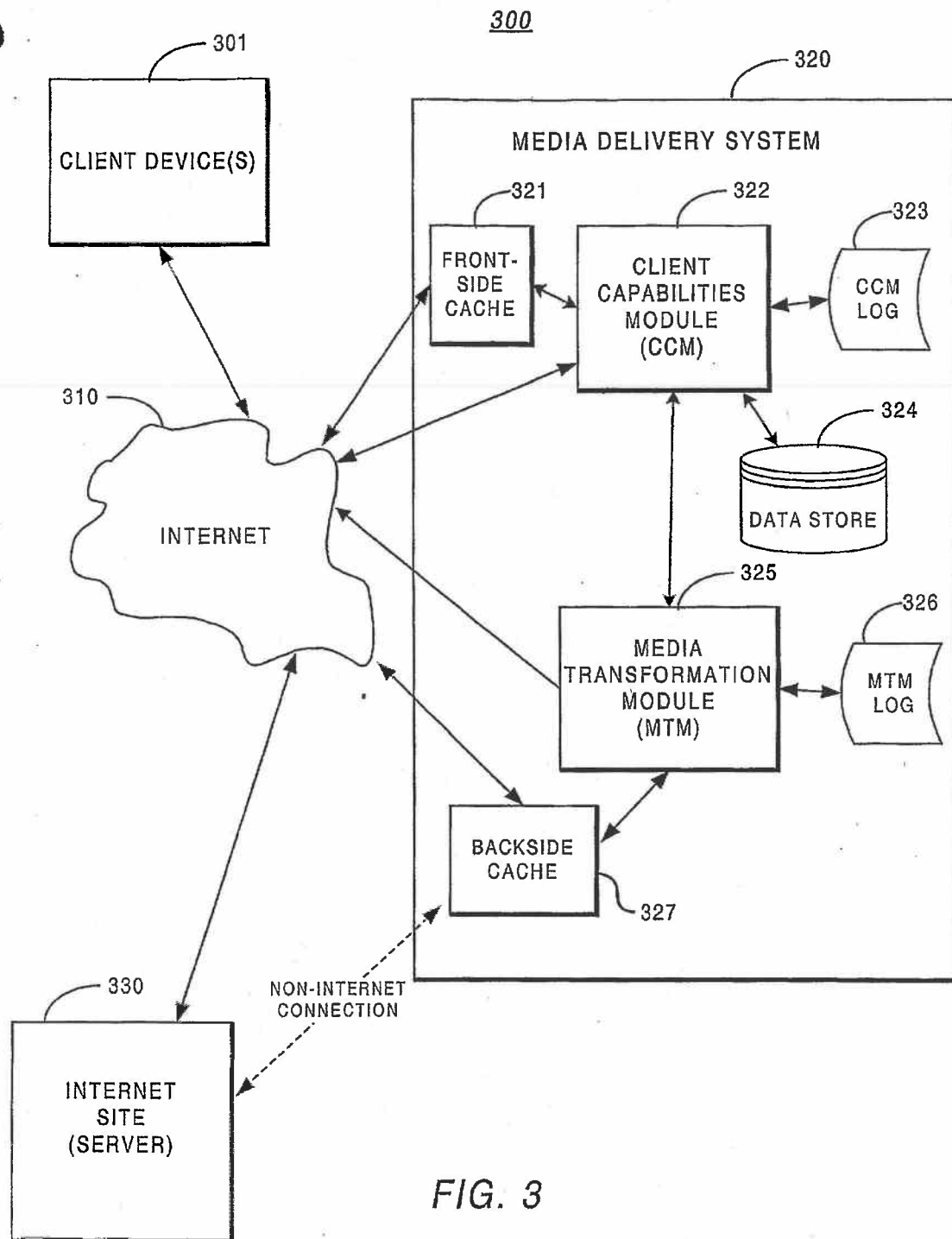


FIG. 3

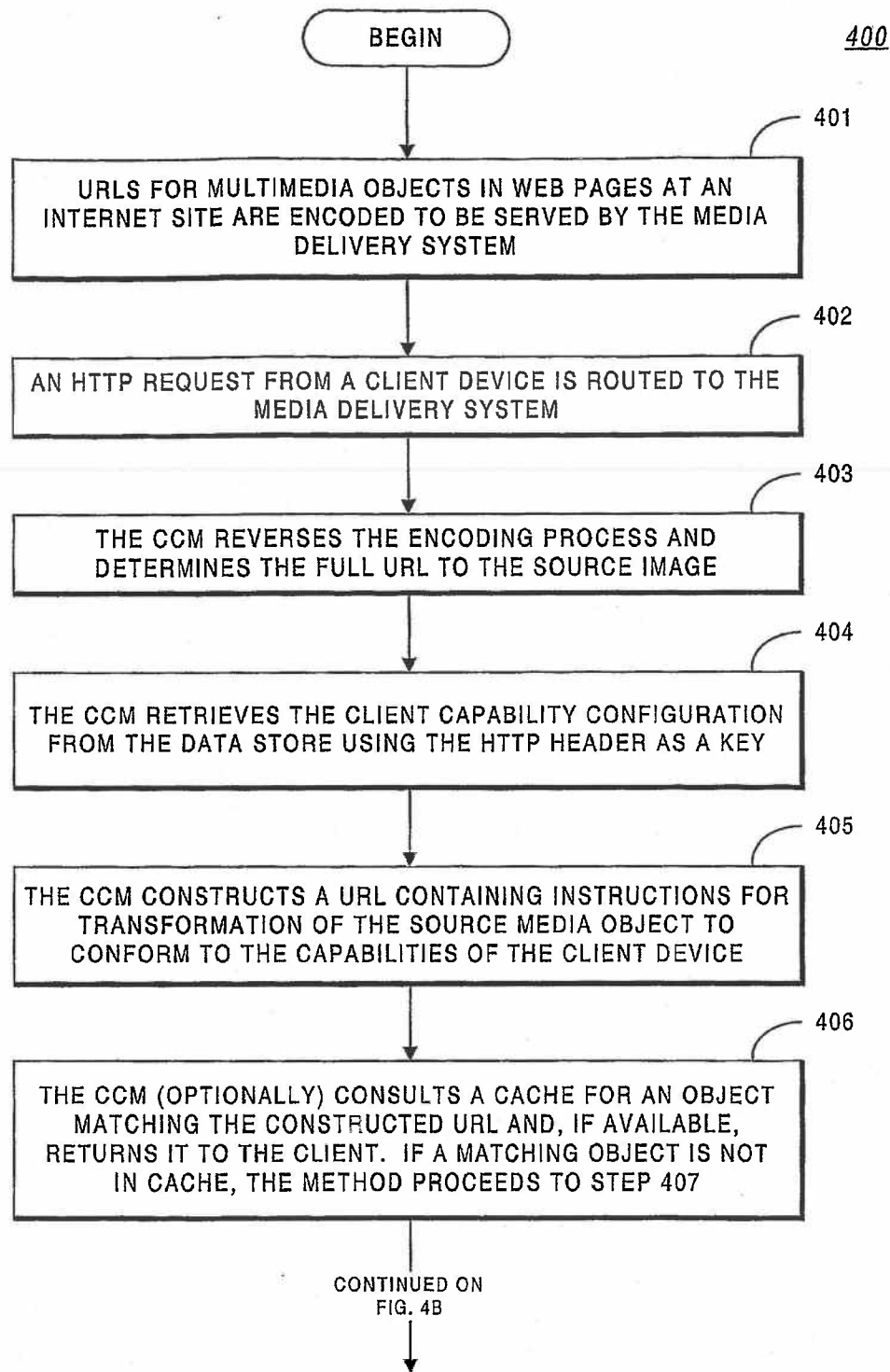


FIG. 4A

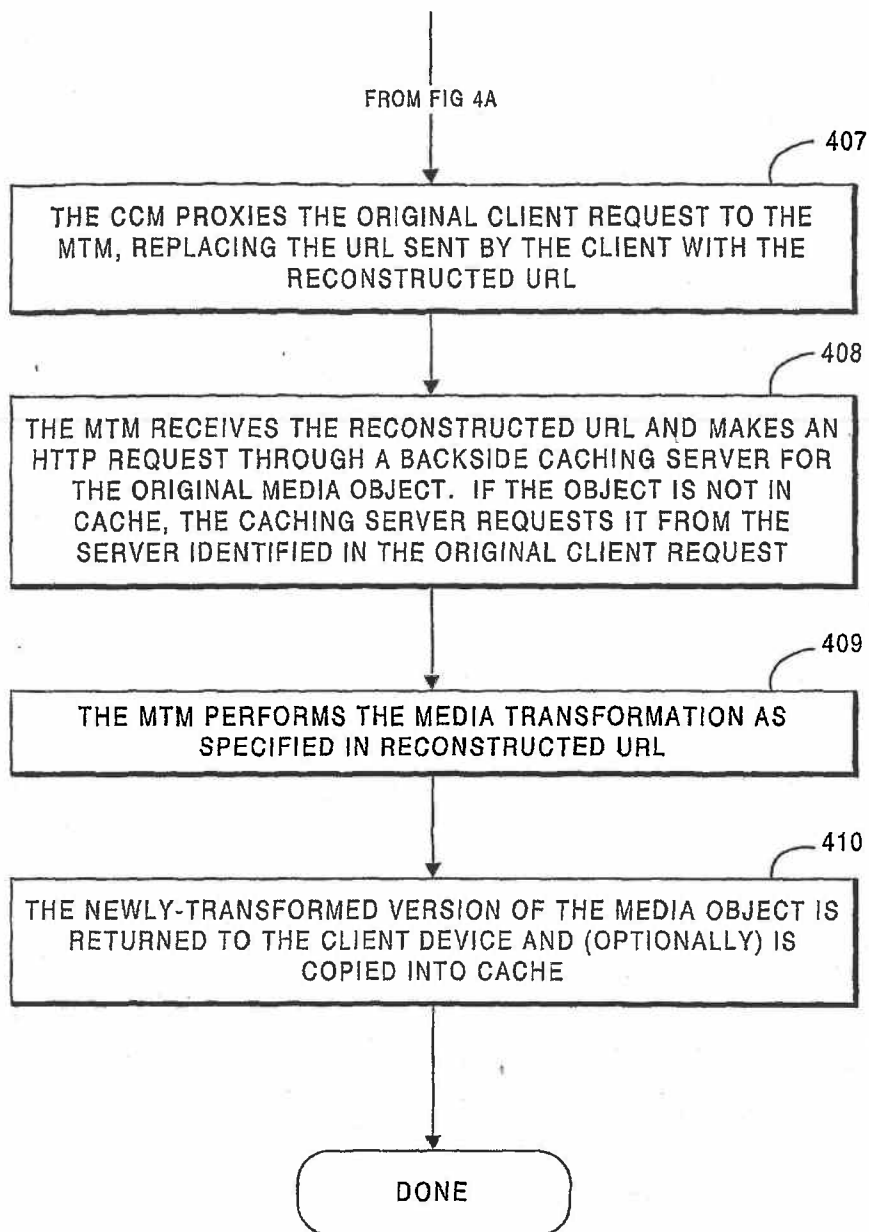


FIG. 4B

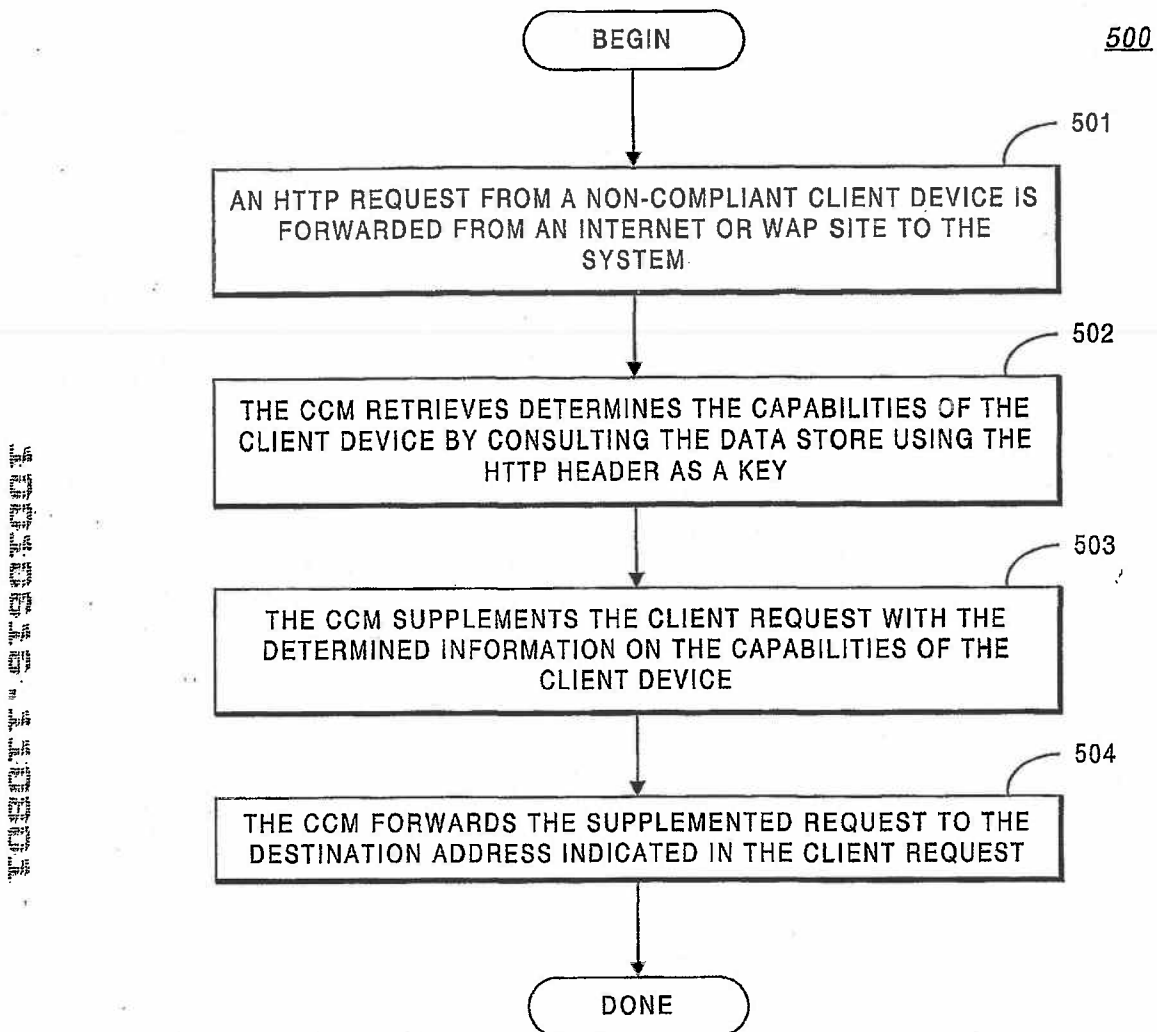


FIG. 5

(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau



(43) International Publication Date
15 May 2003 (15.05.2003)

(10) International Publication Number
WO 03/040893 A2

PCT

(51) International Patent Classification⁷: **G06F**

(21) International Application Number: PCT/US02/36064

(22) International Filing Date:
7 November 2002 (07.11.2002)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/010,616 8 November 2001 (08.11.2001) US

(71) Applicant: **LIGHTSURF TECHNOLOGIES, INC.**
[US/US]; 4th Floor, 110 Cooper Street, Santa Cruz, CA
95060-3901 (US).

(72) Inventors: **MIETZ EGLI, Paul**; 116 Blueberry Drive,
Scotts Valley, CA 95066 (US). **KIRANI, Shekhar**; 109
Washburn Avenue, Capitola, CA 95010 (US). **EASWAR,**
Venkat; 10736 Linda Vista Drive, Cupertino, CA 95014
(US).

(74) Agent: **HICKMAN, Paul, L.**; Perkins Cole LLP, 101 Jef-
ferson Drive, Menlo Park, CA 94025-1114 (US).

(81) Designated States (*national*): AE, AG, AL, AM, AT, AU,
AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU,
CZ, DE, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH,
GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC,
LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW,
MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG,
SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, UZ, VN,
YU, ZA, ZM, ZW.

(84) Designated States (*regional*): ARIPO patent (GH, GM,
KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW),
Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM),
European patent (AT, BE, BG, CH, CY, CZ, DE, DK, EE,
ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, SK,
TR), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ,
GW, ML, MR, NE, SN, TD, TG).

Published:

— *without international search report and to be republished
upon receipt of that report*

*For two-letter codes and other abbreviations, refer to the "Guid-
ance Notes on Codes and Abbreviations" appearing at the begin-
ning of each regular issue of the PCT Gazette.*



WO 03/040893 A2

(54) Title: SYSTEM AND METHODOLOGY FOR DELIVERING MEDIA TO MULTIPLE DISPARATE CLIENT DEVICES
BASED ON THEIR CAPABILITIES

(57) Abstract: An online media delivery system incorporating combining on-the-fly media reformatting with advanced client de-
tection capabilities enabling delivery of appropriate media content to connected client devices is described. The system receives
requests from client devices for media documents or objects, identifies the client device requesting particular media objects from an
HTTP request, determines the media output capabilities of the client device, reformats the source media according to those capabili-
ties and delivers the reformatted media to the client device. The system provides access to media content for multiple disparate client
devices of varying hardware and software capabilities. The system enables delivery of appropriate media content to practically any
device with connectivity capability.

5 **SYSTEM AND METHODOLOGY FOR DELIVERING MEDIA TO MULTIPLE**
 DISPARATE CLIENT DEVICES BASED ON THEIR CAPABILITIES

COPYRIGHT NOTICE

10 A portion of the disclosure of this patent document contains material which is
 subject to copyright protection. The copyright owner has no objection to the facsimile
 reproduction by anyone of the patent document or the patent disclosure as it appears in
 the Patent and Trademark Office patent file or records, but otherwise reserves all
 copyright rights whatsoever.

COMPUTER PROGRAM LISTING APPENDIX

15 A Computer Program Listing Appendix A containing computer program listings is
 included with this application. The disclosure of the Computer Program Listing
 Appendix A is hereby incorporated by reference.

BACKGROUND OF THE INVENTION

20 The present invention relates generally to information processing and, more
 particularly, to an online system providing methodology for improving online access to
 digital media and related information.

 A variety of digital image, digital video and digital audio products are currently
 available to consumers. Regardless of how digital media is recorded, at some later point
 in time, the information must become available to other devices -- that is, become
 available to a larger network of digital devices, so that such information may be displayed
25 on a screen, printed to hard copy, stored, or shared with other people. Today, a large
 number of Web sites exist on the Internet with server computers having the capability to
 organize and display images, video, audio, and other types of media and digital objects.
 In a complementary manner, a multitude of different client devices exist with sufficient
 graphics capabilities for potentially viewing (and/or listening to) this media. For instance,
30 such client devices range from desktop or laptop computers running Web browser
 software to handheld devices (e.g., personal digital assistants running under Palm or
 Windows CE), all connected to the Internet over TCP/IP, each with the capability of
 displaying information.

More recently, a newer class of devices has been introduced with wireless data capabilities as well as display capabilities. These devices connect to the Internet typically over WAP (Wireless Application Protocol). WAP is a communication protocol, not unlike TCP/IP, that was developed by a consortium of wireless companies, including
5 Motorola, Ericsson, and Nokia, for transmitting data over wireless networks. For a description of WAP, see e.g., Mann, S., *The Wireless Application Protocol*, Dr. Dobb's Journal, pp. 56-66, October 1999.

All told, a plethora of graphics-equipped devices exist today that may be connected to the Internet, through both wireless (e.g., 9600 baud) and wireline
10 connections (e.g., 56k baud, DSL, and cable modem). These devices are capable of displaying graphics, including digital images. Recent advances in handheld computing and wireless technologies have enabled manufacturers to embed or include Web browsers in a wide array of devices, including palm-type organizers, wireless phones, and two-way pagers. While all of these Web-enabled clients are capable of at least rudimentary display
15 of text, some also support various multimedia content such as images, video, and sound. Faster wireless Internet access will drive development of client devices capable of playing larger, richer media formats.

However, a fundamental problem exists with current approaches to displaying images and other digital media on many of these devices. Some devices such as personal
20 or laptop computers are capable of receiving and rendering a number of different types of media, including digital images, streaming audio, and streaming video. However, other devices have much more limited capabilities to render digital media. In addition, different devices often support different formats and communication transport protocols.

Consider, for instance, the example of a Palm handheld device. Suppose a user
25 has a "true-color" (e.g., 32-bit depth) 1024 by 768 digital photograph of his or her family on the Web. If the user connects to the Internet using the Palm device, he or she cannot display the photograph because the Palm device may only support four-level or sixteen-level grayscale. Even if the image could be displayed, the transmission time for downloading the image to the Palm device would be impractical. Still further, even if the
30 image could be downloaded, the display for the Palm may be physically too small to render the image in a manner that is acceptable to the user.

This problem is not limited to just image data but also applies to other types of digital objects. Many Internet sites display multi-colored images, streaming video, streaming audio, and other content that is targeted primarily at users with desktop and

laptop computer devices and Web browser software. A desktop or laptop computer has significant processing, storage, and display resources and is capable of rendering large images and video files in multiple colors and formats. On the other hand, a smaller device such as a cellular phone or a personal digital assistant ("PDA") typically does not have the necessary capabilities to display colors or to handle media in particular formats. For example, a cellular phone, which typically has a small screen for display of messages, does not have the software or display capabilities to render a JPEG image. Currently, a user with a PDA or cellphone is typically unable to access much of the information available on many Internet sites.

Certain content providers that are focused primarily on cellular telephone and PDA users do offer Internet sites that display media appropriate for these users. These sites include images with lower resolution and fewer colors in formats supported by these types of devices. However, even content providers focused on cellular telephone and handheld device users have problems supporting the wide number of devices currently in use. Numerous different types of cellular telephones and handheld devices are in use, each having different specifications and capabilities. These devices have different screen sizes, resolution, and color capabilities. Thus, even if a content provider is focused on cellular telephone users, that content provider often must create images and other media offerings geared towards the least capable devices in order to serve the broadest number of users.

Another particular problem in the delivery of media to wireless users is limited bandwidth. In addition to the image display and audio output limitations of many wireless devices, the available bandwidth also limits the ability of these devices to access and use richer media formats. Thus even for devices with better audio and video capabilities, bandwidth considerations may still significantly constrain the types of media that may be supplied to these devices. Even if a user's wireless device is capable of displaying a high-resolution image, he or she may request a lower resolution image because of the time required to download the image given the limited available bandwidth.

In addition to these problems stemming from limited bandwidth and display resources available to many wireless devices, another problem is that present-day Internet services do not attempt to understand the capabilities of particular connected devices. Moreover, even if such Internet services understand some of the capabilities of a particular client device, present-day servers are not designed to act on that information

and transmit information only in the format that is meaningful for such client device. As more and more classes of network-connected devices come to market, this problem can be expected to grow as each device has its own particular characteristics.

- 5 What is needed is a solution that combines on-the-fly media reformatting with advanced client detection to enable the delivery of the appropriate and best possible incarnation of a provider's media to every connected client device. The present invention fulfills this and other needs.
-

GLOSSARY

The following definitions are offered for purposes of illustration, not limitation, in order to assist with understanding the discussion that follows.

5 *Apache:* Apache is a public domain Web server developed by a group of volunteer programmers called the Apache Group. The initial version of the Apache server was developed in 1995 based on the httpd Web server developed at the National Center for Supercomputing Applications, University of Illinois, Urbana-Champaign. Additional information on the Apache Web server and copies of the source code for this Web server are currently available on the Internet at <http://www.apache.org>.

10 *CC/PP:* CC/PP is an abbreviation for *Composite Capabilities/Preference Profiles*, a proposed standard being developed by the World Wide Web Consortium (W3C). A CC/PP profile is a description of device capabilities and user preferences that can be used to guide the adaptation of content presented to that device. The current proposed specification is described by "Composite Capability/Preference Profiles (CC/PP): A user
15 side framework for content negotiation", W3C Note, 27 July 1999, available from the World Wide Web Consortium (W3C). A copy of the proposed standard can be currently found on the Internet at <http://www.w3.org/TR/NOTE-CCPP/>.

20 *HTML:* HTML stands for *HyperText Markup Language*. Every HTML document requires certain standard HTML tags in order to be correctly interpreted by Web browsers. Each document consists of head and body text. The head contains the title, and the body contains the actual text that is made up of paragraphs, lists, and other elements. Browsers expect specific information because they are programmed according to HTML and SGML specifications. Further description of HTML documents is available in the technical and trade literature; see e.g., Ray Duncan, *Power Programming:*
25 *An HTML Primer*, PC Magazine, June 13, 1995.

30 *HTTP:* HTTP stands for *HyperText Transfer Protocol*, which is the underlying communication protocol used by the World Wide Web on the Internet. HTTP defines how messages are formatted and transmitted, and what actions Web servers and browsers should take in response to various commands. For example, when a user enters a URL in his or her browser, this actually sends an HTTP command to the Web server directing it

to fetch and transmit the requested Web page. Further description of HTTP is available in *RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1*. RFC 2616 is available from the World Wide Web Consortium (W3C), and is currently available via the Internet at <http://www.w3.org/Protocols/>. Additional description of HTTP is available in the technical and trade literature; see e.g., William Stallings, *The Backbone of the Web*, BYTE, October 1996.

JPEG: Full-size digital images are maintained in a *Joint Photographic Experts Group* or JPEG format. See e.g., Nelson, M. et al., *The Data Compression Book*, Second Edition, Chapter 11: Lossy Graphics Compression (particularly at pp. 326-330), M&T Books, 1996. Also see e.g., *JPEG-like Image Compression* (Parts 1 and 2), Dr. Dobb's Journal, July 1995 and August 1995 respectively (available on CD ROM as *Dr. Dobb's/CD Release 6* from Dr. Dobb's Journal of San Mateo, CA).

SMTP: SMTP stands for *Simple Mail Transfer Protocol*, a protocol for sending e-mail messages between servers. Most e-mail systems that send mail over the Internet use SMTP to send messages from one server to another; the messages can then be retrieved with an e-mail client using either POP or IMAP. In addition, SMTP is generally used to send messages from a mail client to a mail server.

TCP: TCP stands for *Transmission Control Protocol*. TCP is one of the main protocols in TCP/IP networks. Whereas the IP protocol deals only with packets, TCP enables two hosts to establish a connection and exchange streams of data. TCP guarantees delivery of data and also guarantees that packets will be delivered in the same order in which they were sent. For an introduction to TCP, see, e.g., *RFC 793*.

TCP/IP: Stands for *Transmission Control Protocol/Internet Protocol*, the suite of communications protocols used to connect hosts on the Internet. TCP/IP uses several protocols, the two main ones being TCP and IP. TCP/IP is built into the UNIX operating system and is used by the Internet, making it the de facto standard for transmitting data over networks. For an introduction to TCP/IP, see e.g., *RFC 1180: A TCP/IP Tutorial*. A copy of RFC 1180 is currently available at <ftp://ftp.isi.edu/in-notes/rfc1180.txt>.

UAProf: UAProf or WAG UAProf refers to the proposed *Wireless Access Group User Agent Profile Specification* about how devices such as cellular phones should describe their capabilities to servers. The current proposed specification is described as "WAG

UAProf" (Wireless Application Group User Agent Profile Specification), Wireless Application Protocol Forum, Ltd., Proposed Version 30-May-2001, available from the WAP Forum. A copy of the specification can currently be found on the Internet at <http://www1.wapforum.org/tech/documents/WAP-248-UAProf-20010530-p.pdf>.

- 5 *URL*: Abbreviation of *Uniform Resource Locator*, the global address of documents and other resources on the World Wide Web. The first part of the address indicates what protocol to use, and the second part specifies the IP address or the domain name where the resource is located.

- 10 *WAP*: WAP stands for *Wireless Application Protocol*, which is a communication protocol, not unlike TCP/IP, that was developed by a consortium of wireless companies, including Motorola, Ericsson, and Nokia, for transmitting data over wireless networks. For a description of WAP, see e.g., Mann, S., *The Wireless Application Protocol*, Dr. Dobb's Journal, pp. 56-66, October 1999. More particularly, WAP includes various equivalents of existing Internet protocols. For instance, WML is a WAP version of the
- 15 HTML protocol. Other examples include a WAP browser that is a WAP equivalent of an HTML browser and a WAP gateway (on the server side) that is a WAP equivalent of an HTTP server. At the WAP gateway, HTTP is translated to/from WAP.

- 20 *XML*: Short for *Extensible Markup Language*, a specification developed by the W3C. XML is a pared-down version of SGML, designed especially for Web documents. It allows designers to create their own customized tags, enabling the definition, transmission, validation, and interpretation of data between applications and between organizations. For further description of XML, see, e.g., *Extensible Markup Language (XML) 1.0* specification which is available from the World Wide Web Consortium (www.w3.org). The specification is also currently available on the Internet at
- 25 <http://www.w3.org/TR/REC-xml>.

SUMMARY OF THE INVENTION

The present invention provides an online system incorporating improved methodologies enabling content providers to effectively serve the widest array of clients. It combines on-the-fly media reformatting with advanced client detection capabilities to enable the delivery of the appropriate and best possible incarnation of a provider's media content to every connected client device.

The online media delivery system of the present invention (commercially embodied in eSwitch™-brand media delivery system) receives requests from client devices for media documents or objects, determines the media output capabilities of the device making the request, and serves a transformed version of the media object appropriate for the requesting client device. The system includes a client capabilities module (CCM) and a media transformation module (MTM). These two modules cooperate to identify a client from an HTTP request, determine its media output capabilities, and reformat the source media according to those capabilities. The system also includes a data store containing information on the capabilities of various client devices, an (optional) front-side cache for storing transformed media content, and a backside cache for local storage of original items of content.

The system provides access to media content for multiple disparate client devices -- that is, to target devices of varying hardware and software capabilities. The system enables delivery of appropriate media content to practically any device with connectivity capability. The target devices may include both wireless devices (e.g., cellular phone, handheld personal digital assistant (PDA), and pager) as well as wireline devices (e.g., desktop computer, laptop computer, and videophone).

The improved methodology of the system in providing media content appropriate to a particular device can be summarized as follows. Initially, the URLs of particular full-format multimedia objects on an Internet Web site are modified to be served by the system of the present invention. This is accomplished by modifying the HTML pages on the Web site to replace the URLs of these full-format multimedia objects with URLs which point to the server on which the media delivery system is installed and contains the path to the media objects. The system acts as an HTTP proxy to those original objects, intercepting requests for the original content and serving a transformed version of the content applicable to the requesting client.

When a client device receives user input (e.g., a click) invoking the modified URL for requesting a particular multimedia document, the media output capabilities are communicated to the system by the device or are surmised by the system's client capabilities module. If the device is communicating its capabilities, it does so during the initiation or during every interaction. Alternatively, the device's capabilities are previously stored in the system's data store based on prior knowledge of the device. Based on the information communicated to or surmised by the system, an information record is created describing the target device's capabilities. This information indicates to the system what transformation (if any) is required for translating the original item of media content into a format suitable for the target device.

After the appropriate media format required by the device is determined, the client capabilities module (optionally) checks the front-side cache to see whether the cache already stores a version of the object that has been translated into a format suitable for this particular target device. If the object (transformed for the target device) already exists in the front-side cache, then the client capabilities module may simply return that object to the client device. However, if an appropriate transformed object is not in the cache, then the client capabilities module proceeds to pass the object identifier and the client device parameters on to the media transformation module.

The media transformation module receives requests for a particular item of media content in a particular format from the client capabilities module. The media transformation module obtains the original media object, transforms the object from its original format into the format that is desired for the target device (based on the specified target device capabilities), and returns the converted object to the client device. The media transformation module utilizes a backside cache as an optimization to provide increased efficiency. Media objects are retained in the backside cache to avoid having to retrieve frequently or recently requested items in response to each request. Use of this backside cache reduces the number of calls over the network and expedites conversion and return of media objects to client devices.

BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a block diagram of a computer system in which software-implemented processes of the present invention may be embodied.

5 Fig. 2 is a block diagram of a software system for controlling the operation of the computer system.

Fig. 3 is a block diagram illustrating an online media delivery system of the present invention.

10 Figs. 4A-B comprise a single flowchart illustrating the detailed method steps of the system in determining the capabilities of a target device and transforming and delivering content to such target device in an appropriate format.

Fig. 5 is a flowchart illustrating the operations of the client capabilities module of the present invention in acting as a proxy for incoming HTTP requests from non-compliant devices.

DETAILED DESCRIPTION OF A PREFERRED EMBODIMENT

The following description will focus on the presently-preferred embodiment of the present invention, which is implemented in a desktop application operating in an Internet-connected environment running under a desktop operating system, such as the
5 Microsoft® Windows operating system running on an IBM-compatible PC. The present invention, however, is not limited to any one particular application or any particular environment. Instead, those skilled in the art will find that the system and methods of the present invention may be advantageously embodied on a variety of different platforms, including Macintosh, Linux, BeOS, Solaris, UNIX, NextStep, FreeBSD, and the like.
10 Therefore, the description of the exemplary embodiments that follows is for purposes of illustration and not limitation.

I. Computer-based implementation

A. Basic system hardware (e.g., for desktop and server computers)

The present invention may be implemented on a conventional or general-purpose
15 computer system, such as an IBM-compatible personal computer (PC) or server computer. Fig. 1 is a very general block diagram of an IBM-compatible system 100. As shown, system 100 comprises a central processing unit(s) (CPU) or processor(s) 101 coupled to a random-access memory (RAM) 102, a read-only memory (ROM) 103, a keyboard 106, a printer 107, a pointing device 108, a display or video adapter 104
20 connected to a display device 105, a removable (mass) storage device 115 (e.g., floppy disk, CD-ROM, CD-R, CD-RW, DVD, or the like), a fixed (mass) storage device 116 (e.g., hard disk), a communication (COMM) port(s) or interface(s) 110, a modem 112, and a network interface card (NIC) or controller 111 (e.g., Ethernet). Although not shown separately, a real-time system clock is included with the system 100, in a
25 conventional manner.

CPU 101 comprises a processor of the Intel Pentium® family of microprocessors. However, any other suitable processor may be utilized for implementing the present invention. The CPU 101 communicates with other components of the system via a bi-directional system bus (including any necessary input/output (I/O) controller circuitry and
30 other "glue" logic). The bus, which includes address lines for addressing system memory, provides data transfer between and among the various components. Description

of Pentium-class microprocessors and their instruction set, bus architecture, and control lines is available from Intel Corporation of Santa Clara, CA. Random-access memory 102 serves as the working memory for the CPU 101. In a typical configuration, RAM of sixty-four megabytes or more is employed. More or less memory may be used without departing from the scope of the present invention. The read-only memory (ROM) 103 contains the basic input/output system code (BIOS) -- a set of low-level routines in the ROM that application programs and the operating systems can use to interact with the hardware, including reading characters from the keyboard, outputting characters to printers, and so forth.

Mass storage devices 115, 116 provide persistent storage on fixed and removable media, such as magnetic, optical or magnetic-optical storage systems, flash memory, or any other available mass storage technology. The mass storage may be shared on a network, or it may be a dedicated mass storage. As shown in Fig. 1, fixed storage 116 stores a body of program and data for directing operation of the computer system, including an operating system, user application programs, driver and other support files, as well as other data files of all sorts. Typically, the fixed storage 116 serves as the main hard disk for the system.

In basic operation, program logic (including that which implements methodology of the present invention described below) is loaded from the removable storage 115 or fixed storage 116 into the main (RAM) memory 102, for execution by the CPU 101. During operation of the program logic, the system 100 accepts user input from a keyboard 106 and pointing device 108, as well as speech-based input from a voice recognition system (not shown). The keyboard 106 permits selection of application programs, entry of keyboard-based input or data, and selection and manipulation of individual data objects displayed on the screen or display device 105. Likewise, the pointing device 108, such as a mouse, track ball, pen device, or the like, permits selection and manipulation of objects on the display screen. In this manner, these input devices support manual user input for any process running on the system.

The computer system 100 displays text and/or graphic images and other data on the display device 105. The video adapter 104, which is interposed between the display 105 and the system/s bus, drives the display device 105. The video adapter 104, which includes video memory accessible to the CPU 101, provides circuitry that converts pixel data stored in the video memory to a raster signal suitable for use by a cathode ray tube (CRT) raster or liquid crystal display (LCD) monitor. A hard copy of the displayed

information, or other information within the system 100, may be obtained from the printer 107, or other output device. Printer 107 may include, for instance, an HP LaserJet® printer (available from Hewlett-Packard of Palo Alto, CA), for creating hard copy images of output of the system.

5 The system itself communicates with other devices (e.g., other computers) via the network interface card (NIC) 111 connected to a network (e.g., Ethernet network, Bluetooth wireless network, or the like), and/or modem 112 (e.g., 56K baud, ISDN, DSL, or cable modem), examples of which are available from 3Com of Santa Clara, CA. The system 100 may also communicate with local occasionally-connected devices (e.g., serial
10 cable-linked devices) via the communication (COMM) interface 110, which may include a RS-232 serial port, a Universal Serial Bus (USB) interface, or the like. Devices that will be commonly connected locally to the interface 110 include laptop computers, handheld organizers, digital cameras, and the like.

 IBM-compatible personal computers and server computers are available from a
15 variety of vendors. Representative vendors include Dell Computers of Round Rock, TX, Compaq Computers of Houston, TX, and IBM of Armonk, NY. Other suitable computers include Apple-compatible computers (e.g., Macintosh), which are available from Apple Computer of Cupertino, CA, and Sun Solaris workstations, which are available from Sun Microsystems of Mountain View, CA.

20 **B. Basic system software**

 Illustrated in Fig. 2, a computer software system 200 is provided for directing the operation of the computer system 100. Software system 200, which is stored in system memory (RAM) 102 and on fixed storage (e.g., hard disk) 116, includes a kernel or operating system (OS) 210. The OS 210 manages low-level aspects of computer
25 operation, including managing execution of processes, memory allocation, file input and output (I/O), and device I/O. One or more application programs, such as client application software or “programs” 201 (e.g., 201a, 201b, 201c, 201d) may be “loaded” (i.e., transferred from fixed storage 116 into memory 102) for execution by the system 100.

30 System 200 includes a graphical user interface (GUI) 215, for receiving user commands and data in a graphical (e.g., “point-and-click”) fashion. These inputs, in turn, may be acted upon by the system 100 in accordance with instructions from operating system 210, and/or client application module(s) 201. The GUI 215 also serves to display

the results of operation from the OS 210 and application(s) 201, whereupon the user may supply additional inputs or terminate the session. Typically, the OS 210 operates in conjunction with device drivers 220 (e.g., "Winsock" driver -- Windows' implementation of a TCP/IP stack) and the system BIOS microcode 230 (i.e., ROM-based microcode), particularly when interfacing with peripheral devices. OS 210 can be provided by a conventional operating system, such as Microsoft® Windows 9x, Microsoft® Windows NT, Microsoft® Windows 2000, or Microsoft® Windows XP, all available from Microsoft Corporation of Redmond, WA. Alternatively, OS 210 can also be an alternative operating system, such as the previously-mentioned operating systems.

The above-described computer hardware and software are presented for purposes of illustrating the basic underlying desktop and server computer components that may be employed for implementing the present invention. For purposes of discussion, the following description will present examples in which it will be assumed that there exists a "server" (e.g., Web server) that communicates with one or more "clients" (e.g., media display devices). The present invention, however, is not limited to any particular environment or device configuration. In particular, a client/server distinction is not necessary to the invention, but is used to provide a framework for discussion. Instead, the present invention may be implemented in any type of system architecture or processing environment capable of supporting the methodologies of the present invention presented in detail below.

II. Online rendering of media tailored to capabilities of various devices

A. Introduction

Today, a great volume of various types of media content is available on a multitude of Internet sites. At the same time, a wide range of target devices exist that are capable of displaying (rendering) media to users. These devices include personal computers, laptop computers, personal digital assistants (PDAs), two-way pagers, and cellular telephones. However, several problems exist in the delivery of media content to these devices. Given the vastly different capabilities of the various target devices in use and the different types of images and other media content available on the Internet, content providers currently have problems delivering media content in a manner suitable for display (or rendering) to the user of a particular target device in a satisfactory manner.

One solution for these problems is for an Internet site to display information in a number of different formats. However, this solution requires an Internet content provider to display a large number of copies of the same media object in order to address the various types of devices on the market and their wide range of capabilities. Among the disadvantages of this approach are that it requires the content provider to create, store,
5 and manage multiple pre-rendered versions of the same media in various formats depending on the number of devices to be supported.

The present invention allows a content provider to develop content in one form and deliver the content in multiple forms based on the capabilities of the client device
10 requesting the content. The present invention includes an online system and methodology for providing a client device with media content appropriate for the media output capabilities of the device. The system includes a client capabilities module (CCM) to determine the capabilities of connected client devices and a media transformation (or transcoding) module (MTM) that renders the appropriate media on-the-fly and delivers it
15 to the client device in the appropriate format.

The operations of the present invention can be illustrated by the following example of rendering a digital image to a particular client device. In this example, the original item of media content on an Internet site is a 24-bit color JPEG image and the client device requesting this image is a Palm PDA that supports 16-level grayscale. The
20 client device connects wirelessly to the Internet site and invokes the URL for this JPEG image. The Internet content provider using the present invention has previously revised the Uniform Resource Locator (URL) for this image to refer to the machine on which the client capabilities module of the present invention is installed. As a result, this URL request is routed to the CCM, which identifies the requested image and the client device
25 from this request. The CCM determines the capabilities of the client device in an intelligent fashion by examining the client request to the server to obtain information about the client device and by comparing this information to known device characteristics and capabilities stored in its data store. In this case, the CCM recognizes this device as a specific type of Palm PDA and looks up the device's capabilities in the system's
30 database. Based upon this information, the CCM determines that the JPEG image should be supplied to this Palm device in a 16-level grayscale format.

After the appropriate media format required by the device is determined, the CCM (optionally) checks a front-side cache to see whether the cache already stores a version of the image in the required format. If an appropriate transformed image is not in the cache,

then the CCM requests the image from the media transformation module in a 16-level grayscale format suitable for rendering to this type of Palm PDA device. The MTM obtains the appropriate image, converts it to the appropriate format and serves it to the client device. The system includes intelligence that allows it to optimally translate the
5 images (from their original format) into a format suitable for use by the particular target device. The overall translation or transformation process is performed in a manner that preserves performance and scalability criteria desired for the system.

B. Overview of media delivery system

1. Basic architecture

10 Fig. 3 illustrates an online environment 300 suitable for implementing the present invention. As shown, the environment 300 includes an online media delivery system 320 connected via the Internet (shown at 310) to one or more client devices 301 and at least one Internet site (server) 330. Each of these components will next be described in greater detail.

15 Client device 301 represents one of a variety of target devices (or "clients") that are capable of connecting over the Internet and accessing online content. For example, client devices may include both wireless devices (e.g., cellular phone, handheld PDA (personal data assistant), and pager) as well as wireline devices (e.g., desktop computer, laptop computer, and videophone). Although a single client device is shown in the figure,
20 typically the environment 300 would operate with a multitude of such devices connected.

The Internet server 330 represents a Web server at which items of media content (e.g., audio, video, documents, blob objects, or other items of interest) are stored. During operation, the Internet server 330 typically stores a number of different items of media content that are to be made available to a wide range of client devices. Actual connection
25 between the Internet server 330 and the online media delivery system 320 may occur over the Internet or, optionally, occur via a non-Internet (e.g., WAN) connection as illustrated by the dashed connection line in the figure. In either instance, the Internet server 330 may be housed at the same site as the online media delivery system 320, or may be housed at a remote site, as desired.

30 The online media delivery system 320 functions to detect client device capabilities and, based on that determination, transforms and delivers media content to such devices in appropriate formats to the client device 301. As shown, the media delivery system 320

includes a client capabilities module (CCM) 322, a media transformation module (MTM) 325, and a device capabilities data store 324. As also shown, the client capabilities module 322 is in direct communication with a front-side cache 321 and a CCM log 323; the media transformation module 325, similarly, is in direct communication with backside cache 327 and MTM log 326.

2. Basic operation

During basic system operation, items containing and/or referencing media content (e.g., Web pages) on the Internet server 330 are encoded with a URL that directs clients requesting such items to the system 320. The Internet server 330 may also include the original items of media content, which may be any type of content including digital images, video, audio, documents, "blob" objects, or the like. Alternatively, original items of media content may be stored locally on the system 320 or on another local or remote server to which the system 320 is connected. When a request (e.g., HTTP request) for an item of content is made by the client 301, the request is routed to the client capabilities module 322 of the media delivery system 320. Responsive to the request received from the client device 301, the client capabilities module 322 identifies the (client) device and obtains available information about the device's capabilities. Based on this identification, the client capabilities module 322 retrieves additional information about the capabilities of the client device for displaying or outputting media from the data store 324.

The data store 324 includes media output capabilities of various devices. In the currently preferred embodiment, a corresponding device identifier is employed to index this information. The capabilities stored in data store 324 include information regarding screen resolution, screen color depth, whether images should be rotated to fit on the device's screen display, and other such information as described in more detail below.

The data store 324 is field upgradable so that as new devices are introduced into the market, the profiles of such devices and their capabilities can be added. The client capabilities log 323 includes a record of any client devices that could not be identified or for which capabilities are not available. These log records enable any omitted devices to be identified so that information on these devices can be obtained and added to the data store 324.

After the capabilities of client device 301 have been determined, client capabilities module 322 (optionally) looks into the front-side cache 321, which stores previously converted content, to see if the object is available in the appropriate format. The front-

side cache 321 is an (optional) optimization in which previously converted media objects are retained for supply in response to future requests. The front-side cache 321 may be implemented using least-recently used (LRU) technique to "age out" (i.e., remove) the least-recently used items. If the client capabilities module 322 determines that the
5 appropriate object is not available in the front-side cache 321, it requests the media transformation module 325 to perform an on-the-fly transformation that will supply the object.

The media transformation module (MTM) 325 receives requests for a particular item of media content in a particular format from client capabilities module 322. The
10 media transformation module 325 obtains an original copy of the requested media object, converts it into the requested format, and returns the converted media object to the client device 301. The backside cache 327 is an optimization to provide increased efficiency; it may also be implemented using LRU technique. Original objects retrieved from the Internet server 330 (or another source) are retained in the backside cache 327 to avoid
15 having to retrieve a copy of each requested item in response to each request. Use of this backside cache reduces the number of calls over the network. It also expedites conversion and return of available objects by the media transformation module 325 by avoiding the retrieval of large objects (such as high quality color images) from a remote server.

20 C. Methodology for detecting capabilities of devices and for delivering appropriate media objects

Figs. 4A-B comprise a single flowchart of the detailed method steps of the operations of the system in detecting the capabilities of connected client devices and delivering media objects to such devices in appropriate formats. In step 401, URLs for
25 multimedia objects in Web pages at an Internet site are modified so that such objects will be served by the media delivery system of the present invention. In the currently preferred embodiment, the URLs are prepended with the server name on which the media delivery system is installed. For example, if the subscriber's site contained a logo normally accessed by the URL: *http://www.subscriber.com/img/logo.gif*, the modified
30 URL would be: *http://eswitch.com/www.subscriber.com/img/logo.gif*.

In step 402, an HTTP request from a client device is routed to the media delivery system when a Web page containing these rewritten URLs is opened or the client device selects (clicks on) a rewritten URL. In step 403, the client capabilities module (CCM)

reverses the encoding process performed in step 401 and determines the full URL to the source image. In the currently preferred implementation, this consists of removing the "/eswitch.com" from the request URL.

5 In step 404, the CCM retrieves the client capability configuration from the data store using the HTTP User-Agent header as a key. The configuration information specifies the playback capabilities of the client device, such as display size, color depth, audio channels, and so forth. The configuration information may require examination of additional HTTP request headers to determine the complete capabilities of the client device. Information gathered during this step allows the CCM to understand exactly what
10 capabilities are supported in the target device. In particular, this information indicates to the system what particular transformation operation(s) is required in order to translate the original media object into a format suitable for the target device.

In step 405, the CCM constructs a URL containing commands specific to the media transformation module (MTM). These commands instruct the MTM to transform
15 the source media document or object to conform to the capabilities of the client device that requested the document. This URL points to the MTM server specified in the configuration file. The MTM module can be on the same server or a different server than the CCM. In (optional) step 406, the CCM consults a front-side cache for an object matching the constructed MTM URL. In other words, it looks to see if the front-side
20 cache already stores a version of the media object that has been translated in a manner suitable for this particular target device.

In the currently preferred embodiment, the URL strings used internally within the system are encoded to serve as an index for particular object in a particular format. In this manner, an encoded URL string can indicate that a particular document in a particular
25 format is stored at a particular location. For example the CCM can check for a URL that includes a transformed version of logo.gif with the characteristics: size = 100 pixels, color depth = 8, and color = false. If the document or object (translated for the target device) already exists in the front-side cache, the CCM may simply return the document to the target device. However, if a matching item is not found in the cache, then the method
30 continues as described in step 407.

In step 407, the CCM proxies the original client request, replacing the URL sent by the client with the reconstructed URL created by the CCM. This process is completely transparent to the client: the client device making the request is not informed or aware that the request has been passed on to the MTM. Rather, this transfer is a back-end

process in which the CCM forwards the request made by the client device for fulfillment by the MTM. In step 408, the MTM receives the constructed URL and makes an HTTP request through a backside-caching server for the original media object. If the object is present in the backside cache, it is served from local disk. If not, the caching server
5 requests the object from the Internet site identified in the original URL and caches it for future use. The task of the MTM, at this point, is to transform the media object from its original format into the format that is desired for the target device (based on target device capabilities). In step 409, the MTM performs the media transformation as specified in the reconstructed URL that it received. Once the MTM has carried out this task, in step 410
10 the newly-transformed version of the media object is returned to the client device and (optionally) is also copied into the front-side cache.

D. Use of system to determine and provide information on client capabilities

In addition to serving the role described above, the present invention may also be used to determine the capabilities of client devices and supply this client capabilities
15 information to other systems or devices. For further description about how devices such as cellular phones should describe their capabilities to servers, see, e.g., *WAG UAProf* (*Wireless Application Group User Agent Profile Specification*), Wireless Application Protocol Forum, Ltd., Proposed Version 30-May-2001, available from the WAP Forum. A copy of the specification can currently be found on the Internet at
20 <http://www1.wapforum.org/tech/documents/WAP-248-UAProf-20010530-p.pdf>. A similar specification is described by *Composite Capability/Preference Profiles (CC/PP): A user side framework for content negotiation*, W3C Note, 27 July 1999, available from the World Wide Web Consortium (W3C). A copy of the specification can be currently
found on the Internet at <http://www.w3.org/TR/NOTE-CCPP/>.

25 One or both of these proposed standards may be supported by new devices and device software in the future, but current support for such standards is very limited. Until the device support of these standards is universal, server applications will not be able to take advantage of the standardized device information. This problem can be addressed by configuring the system of the present invention to act as a proxy for incoming HTTP
30 requests from non-compliant devices. When a request is forwarded to the media delivery system with partial or no UAProf or CC/PP information, the client capabilities module looks up the required data and attaches this information to the request before forwarding the request on to its eventual destination. This enables the system to act as a bridge

between the non-compliant client device and those Internet and WAP sites that require compliance with UAProf, CC/PP, or other comparable standards.

Fig. 5 is a flowchart illustrating the operations of the client capabilities module of the media delivery system in acting as a proxy for incoming HTTP requests from non-compliant devices. In step 501, an HTTP request from a non-compliant client device is forwarded from an Internet or WAP site to the system. For these purposes a “non-compliant” client device is one that is not in compliance with UAProf, CC/PP, or similar standards requiring such device to identify its capabilities.

In step 502, the system’s client capabilities module (CCM) retrieves the client capability configuration from the data store using the HTTP User-Agent header as a key. The configuration information specifies the capabilities of the client device, such as display size, color depth, audio channels, and so forth. The configuration information may require examination of additional HTTP request headers to determine the complete capabilities of the client device. Information gathered during this step allows the CCM to understand exactly what capabilities are supported in the target device.

In step 503, the CCM supplements the request made by the particular client device with information regarding the specific capabilities of such client device as illustrated below. In step 504, the CCM returns the supplemented request including details on the client capabilities to the destination specified in such client request.

The following is an example of how the system can be used to append device capabilities information to a request. An example of an incoming request forwarded to the system is as follows:

```

25 GET /index.wml HTTP/1.1
   Host: www.lightsurf.com
   Accept-Charset: ISO-8859-1
   Accept-Language: en
   x-up-subno: pegli_pegli-nt4.office.lightsurf.com
   x-upfax-accepts: none
30 x-up-uplink: none
   x-up-devcap-smartdialing: 1
   x-up-devcap-screendepth: 1
   x-up-devcap-iscolor: 0
   x-up-devcap-immed-alert: 1
35 x-up-devcap-numsoftkeys: 2
   x-up-devcap-screenpixels: 171,108
   x-up-devcap-msize: 8,18
   User-Agent: UP.Browser/3.1-UPG1 UP.Link/3.2

```

As shown, the incoming information reports device capabilities including, for example, color support ("0" or none, for the above device) and screen pixels (171 by 108 pixels for the above device).

Following receipt of the above request, the client capabilities module determines the capabilities of the particular client device in the manner described above. The CCM then attaches this information to the request and forwards the supplemented request on to its eventual destination. A sample request showing the information appended by the CCM is as follows:

```

10  GET /index.wml HTTP/1.1
    Host: www.lightsurf.com
    Accept-Charset: ISO-8859-1
    Accept-Language: en
    User-Agent: UP.Browser/3.1-UPG1 UP.Link/3.2
15  x-wap-profile: http://www.eswitch.com/profiles/0A3F362B.xml

```

In this instance, "0A3F362B.xml" is a generated document containing either the UAProf or CC/PP profile information. A sample UAProf file for this request is as follows:

```

20  <?xml version="1.0"/>
    <RDF xmlns="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
        xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
25  xmlns:prf="http://www.wapform.org/profiles/UAPROF/ccppschem-20010430#">
    <rdf:Description id="WAP-enabled cellular phone">
        <rdf:type
        resource="http://www.wapform.org/profiles/UAPROF/ccppschem-
        20010430#Hardware
        Platform"/>
30  <prf:ScreenSize>171x108</prf:ScreenSize>
        <prf:ColorCapable>No</prf:ColorCapable>
        <prf:NumberOfSoftKeys>2</prf:NumberOfSoftKeys>
        <prf:ScreenSizeChar>8x18</prf:ScreenSizeChar>
        ...
35  </rdf:Description>
    </RDF>

```

The generated document is populated with information from the device's HTTP request and the data store or knowledgebase maintained by the media delivery system. Because the system maintains a knowledgebase of client device characteristics, it is much more capable of creating a complete UAProf or CC/PP document than a server which simply transcodes the information in the HTTP headers.

E. Detailed methods of operation of CCM and MTM modules

1. Client Capabilities Module

The client capabilities module (CCM) identifies a client device from an HTTP request. The CCM uses information supplied in the request, together with media output capability information stored in the data store, to determine the media output capabilities of a particular client device. This enables the CCM to determine the optimal transmission size and playback format for the particular type of client device requesting the item of interest (e.g., media object). For example, an HTTP browser may indicate the browser name (e.g., "Netscape Navigator" or "Microsoft Internet Explorer"), the browser version, and the graphic types it supports. This information is helpful in instances where graphic support is limited, such as a browser running on a set-top box having very limited graphic support (e.g., JPEG support only). In the case that the target device is not able to indicate its capabilities, it will at least indicate its device class, such as a Palm handheld device, a set-top box, a phone with a WAP browser, or the like.

Based on the device class, the CCM obtains information about the capabilities of the device from the device capabilities data store. For example, the device class may be a Palm handheld device of a particular model. Based on this information, the CCM may discern, by looking up the device class in the knowledgebase maintained in the data store, the capabilities of the device, such as display capability (e.g., 16-color), device memory (e.g., 4-8 MB), and display size (e.g., 300 x 500).

The CCM also includes methods to log unidentified clients in the CCM log and to provide notifications at regular intervals to ensure that the knowledgebase is as up-to-date as possible. As new client devices are introduced, the configuration information in the knowledgebase can be updated to add information on these new client devices. The CCM supports either a "push" or a "push/pull" scheme for updating its configuration files. The "push" scheme consists of a secure HTTP POST request or SMTP message sent to the data store containing the replacement configuration file. The "push/pull" scheme consists of sending an HTTP GET request or SMTP message to the data store which causes the server to schedule an update of the local configuration files.

The CCM relies primarily on HTTP headers, particularly the User-Agent header, to identify clients. The CCM also examines information in the protocol layer below HTTP (for example, the origin of the request's IP packets). Once the device has been identified, the CCM consults a hierarchical configuration file (in the data store) which

supplies default values for the device's capabilities and specifies additional headers, such as the Accept header, which can also be used to determine the proper output format for media content. Once the device's capabilities have been determined, the CCM constructs a request to the MTM for the specified media document including the proper reformatting information. The CCM then forwards the client connection to the MTM with the reconstructed request.

In the currently preferred embodiment, the CCM may be implemented as an Apache module with access to the full HTTP request made by the client. The information contained in that request is used to identify the client device making the request through a series of queries against a configuration file. The HTTP User-Agent header is the primary indicator of the requesting client. While User-Agent is an optional header in both HTTP/1.0 and HTTP/1.1, in practice all Web client software send some identifying information in that header on each request. Many clients also send custom headers describing the physical capabilities of their devices. For example, the UP browser (Unwired Planet browser supplied by Openwave Systems, Inc. of Redwood City, CA), which is a WAP browser used in mobile phones, sends out "non-standard" heads in the request. "Non-standard" in this context means that such headers are not covered by the HTTP specification. An example of one of these headers is as follows:

```

20 User-Agent: UP.Browser/3.04-SC02 UP.Link/3.2.3.8
   x-up-devcap-charset: US-ASCII
   x-up-devcap-immed-alert: 1
   x-up-devcap-max-pdu: 1472
   x-up-devcap-msize: 8,10
25 x-up-devcap-numsoftkeys: 2
   x-up-devcap-screenchars: 15,5
   x-up-devcap-screenpixels: 120,50
   x-up-devcap-smartdialing: 1
   x-up-devcap-softkeysize: 7
30 x-up-fax-accepts: none
   x-up-fax-limit: 0
   x-up-subno: RzOyzSrSj-ARAs01_up2.upl.sprintpcs.com
   x-up-uplink: up2.upl.sprintpcs.com

```

The "x-up-devcap-screenpixels" portion of the above header specifically indicates how many pixels in width and height (120 x 50) the client device is capable of displaying. The header shown in the above example contains several details about the capabilities of this particular client device. However, in many cases headers do not include all of these details, and thus the CCM must refer to information stored in the data store to obtain

device characteristics. However, the CCM uses information in the headers like "x-up-devcap-screenpixels" portion of the above header whenever possible.

In the currently preferred implementation, the CCM configuration file (or knowledgebase) in the data store is written in XML to take advantage of that language's hierarchical features. The file consists of a series of <user-agent> entries. An example
 5 <user-agent> entry might appear as follows (line numbers are for reference only):

```

      1 <user-agent header='User-Agent' pattern='UP.Browser' >>
      2 <device-class>wap</device-class>
10     3 <content-type pattern='^image/'>
      4 <capability name='color-depth' default='8' />
      5 <capability name='display-height' default='100'>
      6 <header name='x-up-devcap-screenpixels'
pattern='(\d+),\d+' />
15     7 </capability>
      8 <capability name='display-width' default='100'>
      9 <header name='x-up-devcap-screenpixels'
pattern='\d+,(\d+)' />
20    10 </capability>
      11 <capability name='output-format' default='image/bmp' />
      12 </content-type>
      13 </user-agent>
  
```

In general, tag attribute naming patterns indicate that the values provided to those
 25 attributes will be treated as regular expressions. In the case where information must be extracted from the regular expression, standard "match remember" syntax (parentheses) is used. For example, on line 9 above, the pattern attribute matches the string "120, 50" and indicates that the substring "50" is to be used to set the enclosing capability value.

When the CCM receives a request, it runs through the configuration file, checking
 30 each <user-agent> tag in turn by comparing the value of the HTTP header specified by the header attribute to the string specified in the value attribute. In the majority of cases, the <user-agent> tag is matched against the HTTP User-Agent header, and matching the tag against the HTTP User-Agent header is the default behavior if the header attribute is omitted. Each <user-agent> block is processed in the order it appears in the configuration
 35 file, so <user-agent> tags with more restrictive value attributes appear before <user-agent> tags with less-restrictive value attributes. For example, a <user-agent> tag which reads value='UP.Browser/3.1' is placed before a <user-agent> tag which reads value='UP.Browser'. The <device-class> element is used to separate the different clients into arbitrary groupings based on their capabilities. Some examples of <device-class>
 40 values might be "WAP", "i-mode", "HDML", and "j-phone."

Once a matching <user-agent> tag is found, the CCM determines the MIME type of the media document being requested, generally by issuing an HTTP HEAD request to the document source. Once the MIME type is known, the CCM looks up the appropriate <content-type> block inside the <user-agent> block. In the example above, any request
5 for MIME types that start with "image/" will match the regular expression defined in the pattern attribute of the first <content-type> tag. An actual configuration file may have separate blocks for each supported media type or subtype.

Client capabilities are defined inside the <content-type> tag by one or more <capability> tags. Each media type may require different capabilities. To properly
10 display images, one needs to know display width, height, and color depth. To supply appropriate audio streams, one needs to know bandwidth and whether the device is capable of playing multiple channels. In the simplest case, the configuration file provides previously-stored values for each capability through the mandatory default attribute. Lines 4 and 11 of the above example illustrate this case, setting the color depth and output
15 format for all UP.Browser user agents to 8 bits per pixel and image/bmp, respectively. As described above, some user agents provide additional information to the server in the form of non-standard HTTP headers. The <header> tag can be used inside of a <capability> tag to instruct the CCM to parse these headers and set the device capabilities depending on the header values. In the example, line 6 indicates that the non-standard "x-
20 up-devcap-screenpixels" header should be used, if present, to set the device display width by applying the regular expression provided in the pattern attribute to that header's value. Parentheses are used to isolate a part of the header value to assign to the capability.

Although not specifically shown above, the underlying communication transport may also be inferred from the class or type of device. For example, if the target device is
25 a cellular phone, the system may infer that the underlying communication transport is wireless. As another example, if the target device is a pager that is communicating using WAP, the system may infer that the target device uses wireless communication with limited bandwidth (as compared to a cellular phone). Based on the device class and the incoming request, the system may usually discern whether the communication transport is
30 wireless or wireline. Moreover, very few devices have both wireless and wireline capability. Typical wireline connections include T1, DSL, cable modem and dial-up connections. On the wireless side, typical connections include 9600-baud circuit-switched data call, 9600-baud packet-switched data call, or the newer 64K baud GPR call.

The client capabilities module is also responsible for verifying the source of the original content so that the system can only be used to reformat content of authorized participating sites and not of third parties. Security is enforced by only activating the CCM in response to specific URLs containing the name of a designated server for which content is to be transformed.

The CCM uses the information it derives about the capabilities of a particular client device to construct a request to the media transformation module for the requested media document. This CCM forwards this constructed request, including the proper reformatting information and the client connection to the MTM. The client capabilities module (optionally) implements a front-side cache for transformed media documents. This front-side cache is consulted for transformed document meeting the criteria of the constructed request before the request is forwarded to the MTM. The purpose of this front-side cache is to minimize the load on the MTM module.

2. Media transformation module

The media transformation (or transcoding) module (MTM) accepts HTTP requests for media documents, which contain formatting instructions as request parameters, and reformats the media according to those instructions. The MTM may specialize in a single media type, such as image or video or may support multiple types of media. In basic operation, the media transformation module supports image reformatting by: converting images both to and from the following MIME types: image/jpg, image/bmp, image/gif, image/tiff, image/wbmp, image/iff, image/pcx, and image/png; decreasing image dimensions and increasing image dimensions; supporting rotation of images to conform to the aspect ratio of the client display; and supporting decreasing image color depth and increasing of image color depth. The MTM supports audio reformatting by: converting audio files and streams both to and from the following MIME types: audio/aiff, audio/au, audio/mpeg, audio/wav; and decreasing audio bit rate.

The MTM supports video reformatting by converting video files and streams both to and from the following MIME types: video/mpeg, video/quicktime, video/x-msvideo (AVI), video/x-ms-asf, video/rm, and video/mjpeg. The MTM can also support reformatting of additional multimedia content types and streams as defined by *RFC 2046, Multipurpose Internet Mail Extensions (MIME)*. A copy of RFC 2046 is currently available on the Internet at <http://www.ietf.org/rfc/rfc2046.txt>.

An example of the operation of the media transformation module demonstrating its operation is set forth below. In this example, the MTM receives is translating a JPEG image and receives as input the following:

- 5 Dimensions of output (width and height);
- Type of output device;
- Color space (e.g., RGB or Grayscale);
- Color palette (e.g., True color or indexed); and
- Compression technique.

10

From these inputs, the MTM may output a picture in the specified output format at the specified size. Note that some devices may be characterized differently than their native characteristics. For example a device with a 16-color LCD display may prefer to be characterized as a true-color device. It is then the responsibility of the device to convert the true-color mode image transmitted by the MTM to its internal 16-color mode using internal software/hardware. Any suitable compression scheme may be employed, including proprietary or non-proprietary schemes. Examples include JPEG, JBIG, GIF, or the like. See, e.g., *JPEG-like Image Compression* (Parts 1 and 2), Dr. Dobb's Journal, July 1995 and August 1995 respectively (available on CD ROM as *Dr. Dobb's/CD Release 6* from Dr. Dobb's Journal of San Mateo, CA).

20 The specific operations of the MTM in translating the above-mentioned JPEG image are as follows. First, the input picture is decompressed to generate a bitmap in the color space that was employed. For example, Klikpix uses GUV color space; JPEG supports multiple color space, including YUV and RGB. GUV color space is described in commonly-owned application serial number 09/489,511, filed January 21, 2000; a description of industry-standard color spaces may also be found in that application. The picture is then converted to a "standard" intermediate format, typically in one of the industry-standard color spaces. Examples include:

- 25 L,a,b 16bits/pixel/channel (e.g., used in Adobe Photoshop);
- 30 SRGB 8bits/pixel/channel (e.g., used by Microsoft, HP, and others); and
- YUV

The intermediate format is then mapped to the format required by the output device with the following processing:

1. Image scaling - to scale the image to the desired output size.

2. If only monochrome information is desired, then a monochrome version of the image is generated using standard conversion methods (e.g., using International Telecommunication Union (ITU) recommendations for generating luminance signal Y from R,G,B signals, see, e.g.: ITU-Recommendation BT.601-1 *Encoding parameters of Digital Television for studio*).

3. If the bits/pixel is fewer than 8 - then dithering techniques (such as error diffusion, blue noise masking, or the like - see, e.g., *Recent Progress in Digital Halftoning*, 1994, The Society of Imaging Science and Technology, compiled and edited by Reiner Eschbach, ISBN 0-892080-181-3).

4. If the output device has a color palette (e.g., supporting only 256 colors) then color dithering schemes are used. Such schemes are discussed in some detail in *Computer Graphics-Principles and Practice*, Second Edition, 1990, Foley, van Dam, et al., Addison-Wesley, Reading, MA, ISBN 0-201-12110-7.

5. Compression is optionally performed before data is streamed out. For true-color images, the preferred compression scheme is JPEG. For indexed images GIF, PNG are the preferred methods. For halftoned images, the preferred approach is JBIG compression.

Finally, the generated picture is outputted, and is ready for streaming to a target device for ultimate rendering on that device's display.

An example of the operations of the media transformation module in reformatting an item of media content is shown by the following "AutoRotateOp" function:

```

1: #include "autorotateop.h"
2:
3: AutoRotateOp::AutoRotateOp()
4: {
5: }
6:
7: AutoRotateOp::~AutoRotateOp()
8: {
9: }
10:
11: const char *AutoRotateOp::Name()
12: {
13:     return "autorotate";
14: }
15:
16: const char *AutoRotateOp::Args()
17: {
18:     return "displayWidth(160),displayHeight(120),clockwise(1)";
19: }
20:

```

```

21: const char *AutoRotateOp::Example()
22: {
23:     return "autorotate=118,157";
24: }
5  25:
26: void AutoRotateOp::Process(IMG_image *img)
27: {
28:     int32 w, h, cw;
29:     float displayAspect, photoAspect;
10 30:
31:     GetArg(&w, 160);
32:     GetArg(&h, 120);
33:     GetArg(&cw, 1);
34:
15 35:     displayAspect = (float)w / (float)h;
36:     photoAspect = (float)(img->width) / (float)(img->height);
37:
38:     if ((displayAspect > 1.0f && photoAspect < 1.0f) ||
39:         (displayAspect < 1.0f && photoAspect > 1.0f))
20 40:     {
41:         if (cw)
42:             IMG_RotateRight(img);
43:         else
44:             IMG_RotateLeft(img);
25 45:     }
46: }
47:

```

The above AutoRotateOp function automatically rotates an image to better fit the
 30 available display of a particular device. As shown on line 26 above, the function receives
 a pointer to an image as a parameter. On lines 28 to 36, the display characteristics of the
 device are obtained. The condition on lines 38 to 39 evaluates whether or not the image
 should be presented in portrait orientation (i.e., to display the image vertically) or
 landscape orientation (i.e., to display the image horizontally) to better fit the device
 35 display. Depending on the outcome of this evaluation, a call is made to IMG RotateRight
 or IMG RotateLeft as shown on lines 41 to 44 and the image is rotated either clockwise or
 counterclockwise to fit the display of a particular device. Source code listings providing

further details on the IMG RotateRight and IMG RotateLeft functions are attached hereto as Appendix A.

5 The MTM uses cached versions of the original media objects whenever possible. The MTM caches source media objects in the backside cache to reduce the time required to read the source media and to reduce Internet bandwidth consumption by the media delivery system. This backside caching can be performed by the MTM or by an intermediate reverse proxy cache. The source objects are cached according to the directives specified in their HTTP cache-control headers. In the currently preferred embodiment, an Apache HTTP server configured as a reverse proxy cache is deployed
10 "between" the Internet and the MTM module, so any requests for source multimedia documents are first compared to a local disk cache. The Apache reverse proxy cache module decouples the caching task from the media transformation task for better scalability and maintainability.

15 Appended herewith is Computer Program Listing Appendix A containing source listings, in the C/C++ programming language, providing further description of the present invention. A suitable environment for creating and building C/C++ programs is available from a variety of vendors, including Borland® C++ Builder available from Borland Software Corporation of Scotts Valley, California, and Microsoft® Visual C++ available from Microsoft Corporation of Redmond, WA.

20 While the invention is described in some detail with specific reference to a single-preferred embodiment and certain alternatives, there is no intent to limit the invention to that particular embodiment or those specific alternatives. For instance, those skilled in the art will appreciate that modifications may be made to the preferred embodiment
25 without departing from the teachings of the present invention.

COMPUTER PROGRAM LISTING APPENDIX A

IMGXFORM

```

5  #include "vimage.h"
   #include "imglib.h"

   IMGLIB_API void IMG_FlipH(IMG_image *img)
   {
10      if (img->pixelBytes != 4)
          {
              DEBUG_IMG_FORMAT;
              return;
          }

15      int32 x, y, w;
          uint8 *rowPtr;
          uint32 *pixPtr, *pixPtr2, tempPix;

          w = img->width >> 1;
          rowPtr = img->baseAddr;
          for (y = 0; y < img->height; y++)
          {
20              pixPtr = pixPtr2 = (uint32 *)rowPtr;
              pixPtr2 += img->width - 1;
              for (x = 0; x < w; x++)
              {
25                  tempPix = *pixPtr;
                  *pixPtr++ = *pixPtr2;
                  *pixPtr2-- = tempPix;
              }
              rowPtr += img->rowBytes;
          }
   }

35  IMGLIB_API void IMG_FlipV(IMG_image *img)
   {
          img->baseAddr += img->rowBytes * (img->height - 1);
          img->rowBytes = -img->rowBytes;
   }

40  IMGLIB_API void IMG_RotateLeft(IMG_image *img)
   {
          if (img->pixelBytes != 4)
          {
45              DEBUG_IMG_FORMAT;
              return;
          }

          IMG_image tempImg;

          if (!IMG_DuplicateImage(img, &tempImg))
              return;

          int32 x, y;
          uint8 *rowPtr, *srcRow, *srcPix;
          uint32 *pixPtr;

          if (img->rowBytes < 0)
          {
60              img->baseAddr += (img->height - 1) * img->rowBytes;
              img->rowBytes = -img->rowBytes;
          }
          img->width = tempImg.height;
          img->height = tempImg.width;
          img->rowBytes = img->pixelBytes * img->width;
          rowPtr = img->baseAddr;
          srcRow = tempImg.baseAddr + tempImg.pixelBytes * (tempImg.width -
1);
          for (y = 0; y < img->height; y++)
          {
70              pixPtr = (uint32 *)rowPtr;
              srcPix = srcRow;
              for (x = 0; x < img->width; x++)
              {
75                  *pixPtr = *((uint32 *)srcPix);

```

```

        pixPtr++;
        srcPix += tempImg.rowBytes;
    }
    rowPtr += img->rowBytes;
    srcRow -= tempImg.pixelBytes;
5    }
    IMG_FreeImage(&tempImg);
}

10  IMGLIB_API void IMG_RotateRight(IMG_image *img)
{
    if (img->pixelBytes != 4)
    {
15        DEBUG_IMG_FORMAT;
        return;
    }

    IMG_image tempImg;

20    if (!IMG_DuplicateImage(img, &tempImg))
        return;

    int32 x, y;
    uint8 *rowPtr, *srcRow, *srcPix;
25    uint32 *pixPtr;

    if (img->rowBytes < 0)
    {
30        img->baseAddr += (img->height - 1) * img->rowBytes;
        img->rowBytes = -img->rowBytes;
    }
    img->width = tempImg.height;
    img->height = tempImg.width;
    img->rowBytes = img->pixelBytes * img->width;
35    rowPtr = img->baseAddr;
    srcRow = tempImg.baseAddr + tempImg.rowBytes * (tempImg.height -
1);
    for (y = 0; y < img->height; y++)
    {
40        pixPtr = (uint32 *)rowPtr;
        srcPix = srcRow;
        for (x = 0; x < img->width; x++)
        {
45            *pixPtr = *((uint32 *)srcPix);
            pixPtr++;
            srcPix -= tempImg.rowBytes;
        }
        rowPtr += img->rowBytes;
        srcRow += tempImg.pixelBytes;
50    }
    IMG_FreeImage(&tempImg);
}

55  IMGLIB_API void IMG_Rotate180(IMG_image *img)
{
    IMG_FlipV(img);
    IMG_FlipH(img);
}

60  IMGLIB_API void IMG_Resize(IMG_image *img, int32 newWidth, int32
newHeight, bool highQuality)
{
    if (img->pixelBytes != 4)
    {
65        DEBUG_IMG_FORMAT;
        return;
    }

    if (img->width == newWidth && img->height == newHeight)
70        return;

    IMG_image tempImg;

    if (!IMG_AllocateImage(&tempImg, newWidth, newHeight, img-
75    >imgFormat))
        return;

```

```

//IMG_StretchBlit(img, &tempImg, 0, 0, newWidth, newHeight, true);
    if (highQuality)
    {
5         VImage          vImg;
          vImg.Import(img);
          vImg.Scale(newWidth, newHeight, CImageServer::CUBIC);
          vImg.Export(&tempImg);
    }
10     else
          IMG_StretchBlit(img, &tempImg, 0, 0, newWidth, newHeight,
false);

    IMG_CopyTags(img, &tempImg);
15     IMG_FreeImage(img);
    *img = tempImg;
}

IMGLIB API void IMG_ClampResize(IMG_image *img, int32 maxWidth, int32
20 maxHeight, bool highQuality)
{
    float f, aspect, newAspect;
    int32 width, height;

25     aspect = (float)(img->width) / (float)(img->height);
    newAspect = (float)maxWidth / (float)maxHeight;
    if (aspect > newAspect)
    {
30         width = maxWidth;
        f = (float)width / aspect;
        height = ROUND(f);
        height = CLAMP(height, 1, maxHeight);
    }
    else
35     {
        height = maxHeight;
        f = (float)height * aspect;
        width = ROUND(f);
        width = CLAMP(width, 1, maxWidth);
40     }
    IMG_Resize(img, width, height, highQuality);
}

```

45 Apache Module

```

/**
 * This function creates a device capabilities object
 * and constructs the URL that is passed to the
50 * Media Transcoding Module.
 *
 * @param r
 * @return
 */
55 static int uts_rewrite_uri(request_rec *r) {
    int status;

    // skip if already a proxy
    if (r->proxyreq != NOT_PROXY) {
60         return OK;
    }

    // skip all but GET requests
    if (strcasecmp("GET", r->method) != 0) {
65         return OK;
    }

    // get config file
    uts_dir_config *cfg = (uts_dir_config *)
70     ap_get_module_config(r->per_dir_config, &uts_module);
    uts_server_config *scfg = (uts_server_config *)
    ap_get_module_config(r->server->module_config, &uts_module);

    // (subrequest for content type)
75     string contentType = "image/jpeg";
}

```

```

    /**
     * first, parse the full incoming URI. We're going to treat the
path+args
     * of this URI as the full URI for the proxy request
5     */
    uri_components uc;
    status = ap_parse_uri_components(r->pool, r->uri, &uc);
    if (status != HTTP_OK) {
        ap_log_error(APLOG_MARK, APLOG_ERR | APLOG_NOERRNO, r->server,
10    "Bad input URI: %s", r->uri);
        return DECLINED;
    }

    char *path = ap_pstrdup(r->pool, uc.path);
15
    // pass in headers, get device capabilities back
    DeviceIdentifier::DeviceCapabilities devcap;
    status = dev_ident->getDeviceCapabilities(&devcap, r, contentType);
    if (status != DI_SUCCESS) {
20        ap_log_error(APLOG_MARK, APLOG_ERR | APLOG_NOERRNO, r->server,
        "Device capabilities lookup failed");
        ap_log_error(APLOG_MARK, APLOG_ERR | APLOG_NOERRNO, r->server,
        "Recording headers");
        logHeaders(r);
25        return DECLINED;
    }

    ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, r->server,
30    "%s", devcap.toString().c_str());

    char *sub;
    // pop off initial slash
    sub = ap_getword_nc(r->pool, &path, '/'); // initial slash

35    // if we have an additional subscriber ID, pop it off, too
    if (cfg->subscriber_id_on_url) {
        sub = ap_getword_nc(r->pool, &path, '/'); // subscriber ID
    }

40    //
    // Build option list.
    // TODO: This should really be a function that returns a string.
    //
    char *cfg_clamp, *cfg_mimetype, *cfg_path, *cfg_quality,
45    *cfg_argparam, *cfg_rotate, *cfg_scale, *cfg_filesize ;
    char blank = '\0' ;

    // Clamp or scale?
    if (!devcap.scale_to_width)
50    {
        cfg_clamp = ap_psprintf(r->pool, "%sclampsize=%d&",
            (strchr(cfg->media_transcoder_uri, '?') != NULL ? "&" : "?"),
            (devcap.display_height > devcap.display_width ?
55    devcap.display_height : devcap.display_width));
        cfg_scale = ap_psprintf(r->pool, "%c", blank) ;
    } else {
        cfg_clamp = ap_psprintf(r->pool, "%s",
            (strchr(cfg->media_transcoder_uri, '?') != NULL ? "&" : "?"));
        cfg_scale = ap_psprintf(r->pool, "limitsize=%i,0&",
60    devcap.display_width);
    }

    // Path
    if (path[strlen(path)-2] == '.')
65    {
        switch (path[strlen(path)-1])
        {
            case 'b':
                cfg_path = ap_psprintf(r->pool, "in=\"http://%smp\"&", path)
70    ;
                break ;
            case 'g':
                cfg_path = ap_psprintf(r->pool, "in=\"http://%sif\"&", path)
75    ;
                break ;
            case 'j':

```

```

        cfg_path = ap_psprintf (r->pool, "in=\"http://%spg\"&", path)
;
        break ;
5      case 'p':
        cfg_path = ap_psprintf (r->pool, "in=\"http://%sng\"&", path)
;
        break ;
      case 't':
10      cfg_path = ap_psprintf (r->pool, "in=\"http://%sif\"&", path)
;
        break ;
    } else {
15      cfg_path = ap_psprintf (r->pool, "in=\"http://%s\"&", path) ;
    }

    // Mime Type
    cfg_mimetype = ap_psprintf (r->pool, "outformat=%s&",
20    devcap.mime_type.c_str()) ;

    // Quality
    if (devcap.image_quality != -1)
        cfg_quality = ap_psprintf (r->pool, "outquality=%i&",
25    devcap.image_quality) ;
    else
        cfg_quality = ap_psprintf (r->pool, "outquality=%i&",
        devcap.color_depth) ;

    // Rotate?
    if (devcap.rotate_to_fit)
30      cfg_rotate = ap_psprintf (r->pool, "autorotate=%i,%i,0&",
        devcap.display_width,
        devcap.display_height) ;
    else
35      cfg_rotate = ap_psprintf (r->pool, "%c", blank) ;

    // Arguments
    if (r->args)
40      cfg_argparam = ap_psprintf (r->pool, "%s&", r->args) ;
    else
        cfg_argparam = ap_psprintf (r->pool, "%c", blank) ;

    // Filesize
    if (devcap.maximum_file_size)
45      cfg_filesize = ap_psprintf (r->pool, "filesize=%i&",
        devcap.maximum_file_size) ;
    else
        cfg_filesize = ap_psprintf (r->pool, "%c", blank) ;

50      //
        // End option list build

55      // create URL for lps
        //
        [1]URL[2]Clamp[3]path[4]format[5]quality[6]rotate[7]argparam[8]scale[9]f
        ilesi
        char *url = ap_psprintf (r->pool, "%s%s%s%s%s%s%s%s",
60      cfg->media_transcoder_uri,
        cfg_clamp,
        cfg_path,
        cfg_mimetype,
        cfg_quality,
        cfg_rotate,
65      cfg_argparam,
        cfg_scale,
        cfg_filesize) ;

    ap_log_error(APLOG_MARK, APLOG_DEBUG | APLOG_NOERRNO, r->server,
70    "proxying: %s", url);

    // double-check that this is a valid URL
    status = ap_parse_uri_components(r->pool, url, &uc);
    if (status != HTTP_OK) {
75      ap_log_error(APLOG_MARK, APLOG_ERR | APLOG_NOERRNO, r->server,
        "Bad proxy URI: %s", url);
    }

```

```

    }
    return DECLINED;
}

// internal redirect won't work for remote servers, so
// set this up as a proxy
/* example of working request member variables
r->proxyreq    = PROXY_PASS;
r->uri         = "/lsps";
r->filename     = "proxy:http://pegli-
10 nt4.office.lightsurf.com:10104/lsps";
r->args        =
"clampsiz=100&in=http://www.lightsurf.com/images/misc/pk_sunset.jpg";
r->handler     = "proxy-server";
*/
15
r->proxyreq    = PROXY_PASS;
r->uri         = ap_pstrdup(r->pool, uc.path);
r->filename     = ap_pstrcat(r->pool, "proxy:", uc.scheme, "://",
uc.hostinfo, uc.path, NULL);
20 r->args        = ap_pstrdup(r->pool, uc.query);
r->handler     = "proxy-server";

return DECLINED;
}
25

```

DeviceIdentifier

```

30 /**
   * DeviceIdentifier.cpp
   * implementation of DeviceIdentifier and DeviceCapabilities classes
   * 4/23/2001 pae
   */
35

#include <string>
#include <sstream.h>
#include <stdlib.h>
40

#include "httpd.h"
#include "http_log.h"

#include "Regex.hpp"
45 include "DeviceIdentifier.hpp"

static Regex regex;

50 DeviceIdentifier::DeviceIdentifier(const char * filename) {
    try {
        config = new XMLConfigFile(filename);
    }
    catch (...) {
55 }
}

DeviceIdentifier::DeviceIdentifier() {
    config = new XMLConfigFile("/lsurf/uts/conf/devices.xml");
60 }

DeviceIdentifier::~DeviceIdentifier() {
    if (config != NULL) {
65 }
    delete config;
}

/**
70 * Retrieve a named capability value from the configuration
   * file. This function will first find the <capability>
   * node identified by <code>capabilityName</code>, then
   * determine if any <header> tags apply. The
   * value is returned as a string, which can then be
75 * converted to the appropriate type.
   */

```

```

    * @param r
    * @param baseXPath
    * @param capabilityName
    * @return
5    */
    string DeviceIdentifier::getCapability(request_rec *r, string baseXPath,
    string capabilityName) {

        /**
10       * The device capabilities data storage is implemented as an XML
        file.
        * A member variable named "config" allows us to make XPath queries
        * against the XML file.
        */
15       string cquery =
        baseXPath +
        string("/capability[@name=\"" +
        capabilityName +
        string("\"]");

20       string value;

        // first, look through headers (if any)
        XMLConfigFile::StringVectorType headernames = config->query(cquery +
25       string("/header/@name"));
        for (unsigned int i=0; i < headernames.size(); i++) {
            const char *headerval = ap_table_get(r->headers_in,
            headernames[i].c_str());
            if (headerval != NULL) {
30               // found a matching header, so grab the pattern and run the
                regex
                XMLConfigFile::StringVectorType patterns = config-
                >query(cquery + string("/header[@name=\"" + headernames[i] +
35               string("\"]/@pattern"));
                if (regex.group(patterns[0], string(headerval), &value)) {
                    // yes, there is a match between the <header pattern="">
                    and the header value
                    if (value.size() > 0) {
40                       // value now contains the matched substring
                        break;
                    } else {
                        // there was a pattern match, but no parenthesized
                        substring, so use the value of the "default" attribute
                        XMLConfigFile::StringVectorType defaults = config-
45                       >query(cquery + string("/header[@name=\"" + headernames[i] +
                        string("\"]/@default"));
                        value = defaults[0];
                    }
                }
50             }
        }

        // if we didn't get anything from the headers, use the default value
        XMLConfigFile::StringVectorType defaults = config->query(cquery +
55       string("/@default"));
        if (defaults.size() > 0) {
            value = defaults[0];
        }

60       // TODO: if there isn't a default, die a horrible death
        // this may not be necessary if we publish a schema for the config
        file

        return value;
65     }

    /**
70     * Convert the result of <code>getCapability()</code>
    * to an integer.
    *
    * @param r
    * @param baseXPath
75     * @param capabilityName
    * @return

```



```

    */
    int DeviceIdentifier::getCapabilityInt(request_rec *r, string baseXPath,
    string capabilityName) {
        string value = getCapability(r, baseXPath, capabilityName);
5         if (value.size() > 0) {
            return atoi(value.c_str());
        } else {
            return -1;
10     }
}

/**
15  * Convert the result of <code>getCapability()</code>
    * to a boolean.
    *
    * @param r
    * @param baseXPath
20  * @param capabilityName
    * @return
    */
    int DeviceIdentifier::getCapabilityBool(request_rec *r, string
    baseXPath, string capabilityName) {
25         string value = getCapability(r, baseXPath, capabilityName);
        return (!value.compare("true") || !value.compare("1")); // anything
        but true or 1 is considered false
    }

30

/**
    * Fill in a DeviceCapabilites structure based on the
    * HTTP request headers and the content type of the
35  * requested document. The main job of this function
    * is to find the correct node in the config file for
    * the given User-Agent header, then call
    * <code>getCapabilityStr()</code>, <code>getCapabilityInt()</code>,
    * and <code>getCapabilityBool()</code> to fill in
40  * the device capabilities.
    *
    * @param devcap
    * @param r
    * @param contentType
45  */
    int
    DeviceIdentifier::getDeviceCapabilities(DeviceIdentifier::DeviceCapabili
    ties *devcap, request_rec *r, string contentType) {
50         if (devcap == NULL) {
            // sorry, no can do
            ap_log_error(APLOG_MARK, APLOG_ERR, r->server, "Cannot call
            getDeviceCapabilities() with a null DeviceCapabilities parameter");
55         }
        return DI_ERROR;

        XMLConfigFile::StringVectorType xpath_results;
        string base_xpath_query;

60         // find the correct user-agent node
        const char *ua_header = ap_table_get(r->headers_in, "User-Agent");

        xpath_results = config->query("//user-agent/@pattern");
        int found = FALSE;
65         for (unsigned int i=0; i < xpath_results.size(); i++) {
            if (regex.match(xpath_results[i], string(ua_header))) {
                base_xpath_query.append("//user-agent[@pattern=\"");
                base_xpath_query.append(xpath_results[i]);
                base_xpath_query.append("\"]");
70                 found = TRUE;
                break;
            } else if (regex.last_error.size() > 0) {
                // this will complain a lot if there are regex errors
                ap_log_error(APLOG_MARK, APLOG_ERR, r->server, "Regular
75 expression error: %s", regex.last_error.c_str());
            }
        }
    }
}

```



```

        ap_log_error(APLOG_MARK, APLOG_DEBUG, r->server, "Regex:
%s", xpath_results[i].c_str());
    }
5     if (!found) {
        ap_log_error(APLOG_MARK, APLOG_DEBUG, r->server, "Could not find
configuration entry for User-Agent \"%s\"", ua_header);
        // TODO: log or send email
        return DI_NOTFOUND;
10    }

    // within that node, find the correct mime-type node
    xpath_results = config->query(base_xpath_query + string("/mime-
type/@pattern"));
15    found = FALSE;
    for (unsigned int i=0; i < xpath_results.size(); i++) {
        if (regex.match(xpath_results[i], contentType)) {
            base_xpath_query.append("/mime-type[@pattern=\"%s\"]");
            base_xpath_query.append(xpath_results[i]);
            base_xpath_query.append("\"]");
            found = TRUE;
            break;
        } else if (regex.last_error.size() > 0) {
25            // again, here's another big complainer
            ap_log_error(APLOG_MARK, APLOG_ERR, r->server, "Regular
expression error: %s", regex.last_error.c_str());
            ap_log_error(APLOG_MARK, APLOG_DEBUG, r->server, "Regex:
%s", xpath_results[i].c_str());
30        }
    }
    if (!found) {
        ap_log_error(APLOG_MARK, APLOG_DEBUG, r->server, "Could not find
<mime-type> entry in config file for document MIME type \"%s\"",
35        contentType.c_str());
        // TODO: log or send email
        return DI_NOTFOUND;
    }

40    ap_log_error(APLOG_MARK, APLOG_DEBUG, r->server, "Successfully
identified device, User-Agent: \"%s\"", ua_header);

    // fill in the DeviceCapabilities structure
    // note defaults are: string="", int=-1, bool=false
45    devcap->mime_type = getCapability(r, base_xpath_query,
"output-mime-type");
    devcap->display_width = getCapabilityInt(r, base_xpath_query,
"display-width");
    devcap->display_height = getCapabilityInt(r, base_xpath_query,
50    "display-height");
    devcap->color_depth = getCapabilityInt(r, base_xpath_query,
"color-depth");
    devcap->image_quality = getCapabilityInt(r, base_xpath_query,
"image-quality");
55    devcap->color_capable = getCapabilityBool(r, base_xpath_query,
"color-capable");
    devcap->rotate_to_fit = getCapabilityBool(r, base_xpath_query,
"rotate-to-fit");
    devcap->scale_to_width = getCapabilityBool(r, base_xpath_query,
60    "scale-to-width");
    devcap->palette = getCapability(r, base_xpath_query,
"palette");
    devcap->maximum_file_size = getCapabilityInt(r, base_xpath_query,
"maximum-size");
65    devcap->video_frame_rate = getCapabilityInt(r, base_xpath_query,
"video-frame-rate");
    devcap->audio_encoding_rate = getCapabilityInt(r, base_xpath_query,
"audio-encoding-rate");
    devcap->audio_channels = getCapabilityInt(r, base_xpath_query,
70    "audio-channels");

    return DI_SUCCESS;
};
75

```

```

/*****
*****/

5 // utility functions to print out the DeviceCapabilities object
void DeviceIdentifier::DeviceCapabilities::print(ostream *os) {
    *os << this->toString();
};

10 string DeviceIdentifier::DeviceCapabilities::toString() {
    std::ostringstream buffer;
    buffer << "DeviceCapabilites {"
    << " mime-type: " << mime_type
    << "; display width: " << display_width
15 << "; display height: " << display_height
    << "; color depth: " << color_depth
    << "; color capable? " << (color_capable ? "yes" : "no")
    << "; rotate to fit? " << (rotate_to_fit ? "yes" : "no")
    << "; video frame rate: " << video_frame_rate
20 << "; audio encoding rate: " << audio_encoding_rate
    << "; audio channels: " << audio_channels
    << "}"
    << "\0";
    return buffer.str();
25 };

ostream &operator<<(ostream &os, DeviceIdentifier::DeviceCapabilities
&devcap) {
    devcap.print(&os);
30 return os;
};

```

WHAT IS CLAIMED IS:

1. In an online system, a method for determining the capabilities of client devices and supplying media content in a format suitable for such devices, the method comprising:

- 5 receiving a request to provide a target device with a copy of a particular media object;
- determining capabilities of the target device;
- based on the capabilities of the target device, determining a format that is desired for providing the target device with a copy of the media object;
- 10 translating the particular media object into a copy having said determined format; and providing the target device with the copy having said determined format.

2. The method of claim 1, further comprising storing the copy having said determined format in a cache memory.

3. The method of claim 2, further comprising:
- 15 receiving from a target device a subsequent request for the particular object in the determined format; and
- providing the target device with the copy stored in said cache memory.

4. The method of claim 1, further comprising:
- obtaining a copy of said particular media object from a connected server for
- 20 translation of said media object.

5. The method of claim 4, further comprising:
- storing in cache memory a cached copy of said media object received from said connected server; and
- in response to subsequent requests for translation of said media object, using the
- 25 copy of said media object stored in cache memory.

6. The method of claim 1, wherein the capabilities of the target device include screen resolution.

7. The method of claim 1, wherein the capabilities of the target device include screen size.

8. The method of claim 1, wherein the capabilities of the target device include color support.

5 9. The method of claim 1, wherein the capabilities of the target device include bit rate.

10. The method of claim 1, wherein the capabilities of the target device include currently-available communication medium that the target device employs to transmit its request.

10 11. The method of claim 10, wherein currently-available communication medium comprises wireless communication.

12. The method of claim 10, wherein currently-available communication medium comprises wireline communication.

15 13. The method of claim 1, wherein said step of determining capabilities of the target device includes examining the request submitted by the device

14. The method of claim 1, wherein said step of determining capabilities of the target device includes examining the HTTP header submitted by the device.

15. The method of claim 14, wherein examining the HTTP header submitted by the device includes examining the HTTP User-Agent header.

20 16. The method of claim 1, wherein said step of determining capabilities of the target device includes querying the device for its capabilities.

17. The method of claim 1, wherein said step of determining capabilities of the target device includes determining capabilities from a knowledgebase, based on a device class for the target device.

25 18. The method of claim 17, further comprising:

recording a log record of target devices that are not recognized to enable the capabilities of said devices to be added to the knowledgebase.

19. The method of claim 18, further comprising:
automatically issuing notifications regarding said target devices that are not
5 recognized.

20. The method of claim 1, wherein said step of determining a format that is desired includes determining an appropriate resolution for rendering the particular image at the target device.

21. The method of claim 1, wherein said step of determining a format that is
10 desired includes determining an appropriate color space for rendering a particular image at the target device.

22. The method of claim 1, wherein said step of determining a format that is desired includes determining an appropriate image size for rendering the particular image at the target device.

15 23. The method of claim 1, wherein said step of determining a format that is desired includes determining whether to rotate the particular image to conform to the aspect ratio of the target device display.

24. The method of claim 1, wherein said step of determining a format that is desired includes determining the appropriate bit rate for the target device.

20 25. The method of claim 1, wherein said step of determining a format that is desired includes determining communication bandwidth available for transmitting a copy of the particular media object to the target device.

26. The method of claim 25, wherein the communication bandwidth available is determined, at least in part, based on the HTTP request header received from the target
25 device.

27. The method of claim 25, wherein the communication bandwidth available is determined, at least in part, based on a device class for the target device.

28. The method of claim 1, wherein said target device includes a handheld computing device having display capability.

29. The method of claim 1, wherein said target device includes a handheld computing device having digital audio capability.

5 30. The method of claim 1, wherein said target device includes a cellular phone device having display capability.

31. The method of claim 1, wherein said target device includes a cellular phone device having digital audio capability.

10 32. The method of claim 1, wherein said target device includes a pager device having display capability.

33. The method of claim 1, wherein said target device includes a personal computer having display capability.

34. The method of claim 1, wherein said target device includes a personal computer having digital audio capability.

15 35. The method of claim 1, wherein said target device includes WAP (Wireless Application Protocol) support.

36. The method of claim 1, wherein said media objects include digital images.

37. The method of claim 1, wherein said digital objects include digital video.

38. The method of claim 1, wherein said digital objects include digital audio.

20 39. An online system for providing digital media to target devices, the system comprising:

 a capabilities module for determining the capabilities of a particular target device;

 a transformation module for:

 automatically retrieving a copy of a particular media object; and

providing the target device with a copy of said object, said copy being automatically translated into a particular format based on the capabilities of the target device.

5 40. The system of claim 39, further comprising:
a cache memory for storing copies of media objects that have been translated.

41. The system of claim 40, wherein said system first attempts to satisfy the request by retrieving a copy of the particular object in the particular format from the cache memory.

10 42. The system of claim 39, further comprising:
a cache memory for storing copies of media objects that have been retrieved.

43. The system of claim 42, wherein said system first attempts to retrieve a copy of the particular media object from the cache memory before retrieving a copy from a remote server.

15 44. The system of claim 39, wherein each digital object stored by said system is identified by a unique uniform resource locator (URL).

45. The system of claim 44, wherein said unique URL is encoded with the characteristics of said digital object.

46. The system of claim 44, wherein said unique URL includes the color depth of said digital object.

20 47. The system of claim 44, wherein said unique URL includes the image size of said digital object.

48. The system of claim 44, wherein said unique URL includes the resolution of said digital object.

25 49. The system of claim 44, wherein said unique URL includes the bit rate of said digital object.

50. The system of claim 44, wherein said system stores URLs for each of the digital objects, and wherein the capabilities module is capable of determining the particular digital object that may be provided to the target device from said URLs.

5 51. The system of claim 39, wherein the capabilities of the target device include screen resolution.

52. The system of claim 39, wherein the capabilities of the target device include screen size.

53. The system of claim 39, wherein the capabilities of the target device include color support.

10 54. The system of claim 39, wherein the capabilities of the target device include bit rate.

55. The system of claim 39, wherein the capabilities of the target device include currently-available communication medium that the target device employs to transmit its request.

15 56. The system of claim 55, wherein currently-available communication medium comprises wireless communication.

57. The system of claim 55, wherein currently-available communication medium comprises wireline communication.

20 58. The system of claim 39, wherein said capabilities module includes the ability to determine the capabilities of the target device from its HTTP header.

59. The system of claim 58, wherein said capabilities module includes the ability to determine the capabilities of the target device from its HTTP User-Agent header.

60. The system of claim 39, wherein said capabilities module includes the ability to query the target device for its capabilities.

61. The system of claim 39, wherein said capabilities module includes a knowledgebase for determining the capabilities of the target device based on its device class.

62. The system of claim 61, further comprising:

5 a log record for recording target devices that are not recognized to enable the capabilities of said devices to be added to the knowledgebase.

63. The system of claim 39, wherein said particular format is selected based on an appropriate resolution for rendering the particular media object at the target device.

10 64. The system of claim 39, wherein said particular format is selected based on an appropriate color space for rendering the particular media object at the target device.

65. The system of claim 39, wherein said particular format is selected based on an appropriate image size for rendering the particular media object at the target device.

66. The system of claim 39, wherein said particular format is selected based on an appropriate bit rate for rendering the particular media object at the target device.

15 67. The system of claim 39, wherein said particular format is selected based on communication bandwidth available for transmitting a copy of the particular media object to the target device.

68. The system of claim 67, wherein the communication bandwidth available is determined, at least in part, based on a device class for the target device.

20 69. The system of claim 39, wherein said target device includes a handheld computing device having display capability.

70. The system of claim 39, wherein said target device includes a handheld device having digital audio capability.

25 71. The system of claim 39, wherein said target device includes a cellular phone device having display capability.

72. The system of claim 39, wherein said target device includes a cellular phone device having digital audio capability.

73. The system of claim 39, wherein said target device includes a pager device having display capability.

5 74. The system of claim 39, wherein said target device includes a personal computer having display capability.

75. The system of claim 39, wherein said target device includes a personal computer device having digital audio capability.

10 76. The system of claim 39, wherein said target device includes WAP (Wireless Application Protocol) support.

77. In an online system, a method for determining the capabilities of client devices, the method comprising:

receiving an original request from a target device in which said target device does not include information regarding its capabilities;

15 determining capabilities of the target device;

supplementing said original request received from said target device with information about the capabilities of said target device; and

forwarding said supplemented request to a destination specified in said original request.

20 78. The method of claim 77, wherein said step of determining capabilities of the target device includes examining the request submitted by the device

79. The method of claim 77, wherein said step of determining capabilities of the target device includes examining the HTTP header submitted by the device.

25 80. The method of claim 79, wherein examining the HTTP header submitted by the device includes examining the HTTP User-Agent header.

81. The method of claim 77, wherein said step of determining capabilities of the target device includes querying the device for its capabilities.

82. The method of claim 77, wherein said step of determining capabilities of the target device includes determining capabilities from a knowledgebase, based on a device class for the target device.

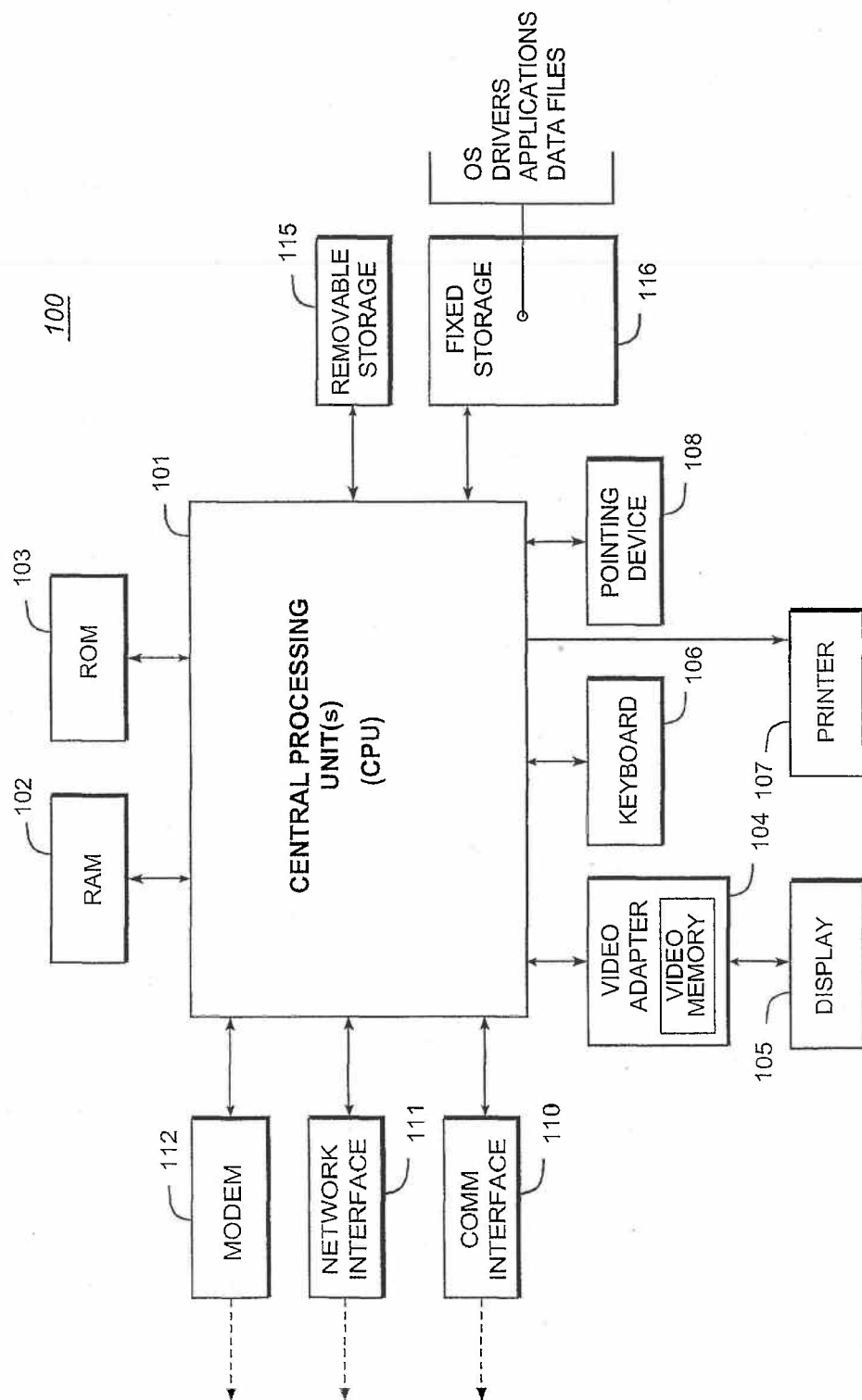


FIG. 1
(PRIOR ART)

.2/6

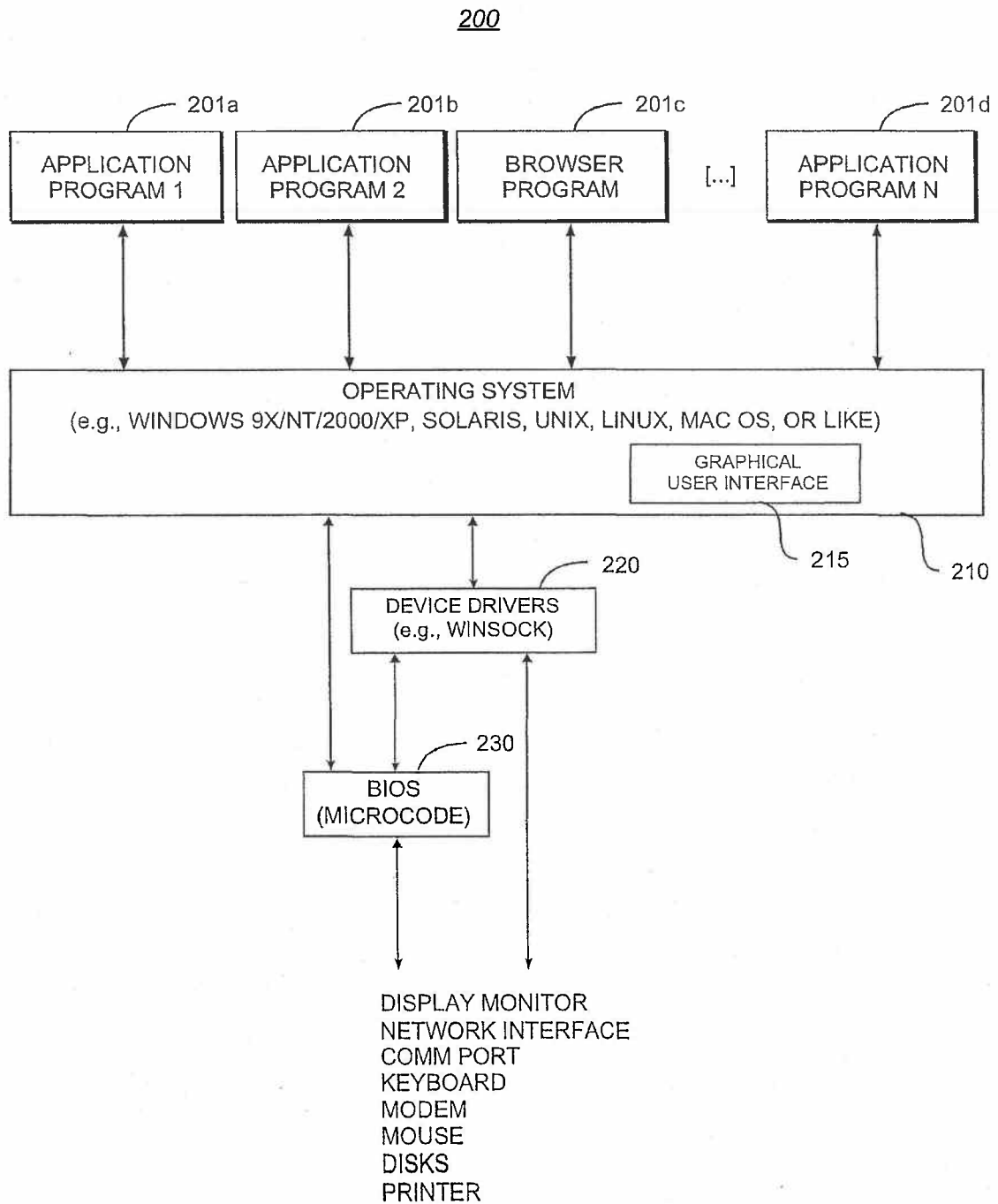


FIG. 2

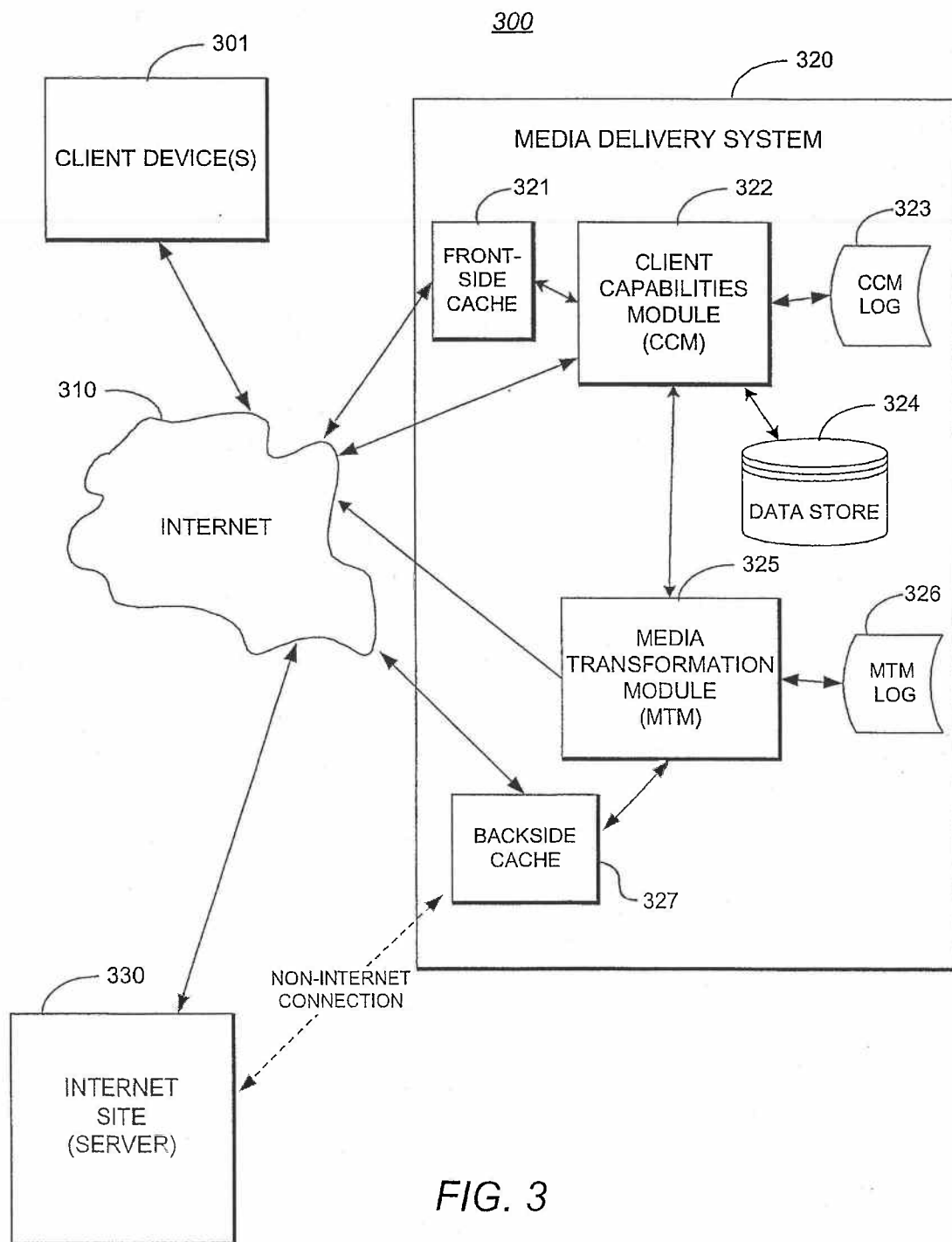


FIG. 3

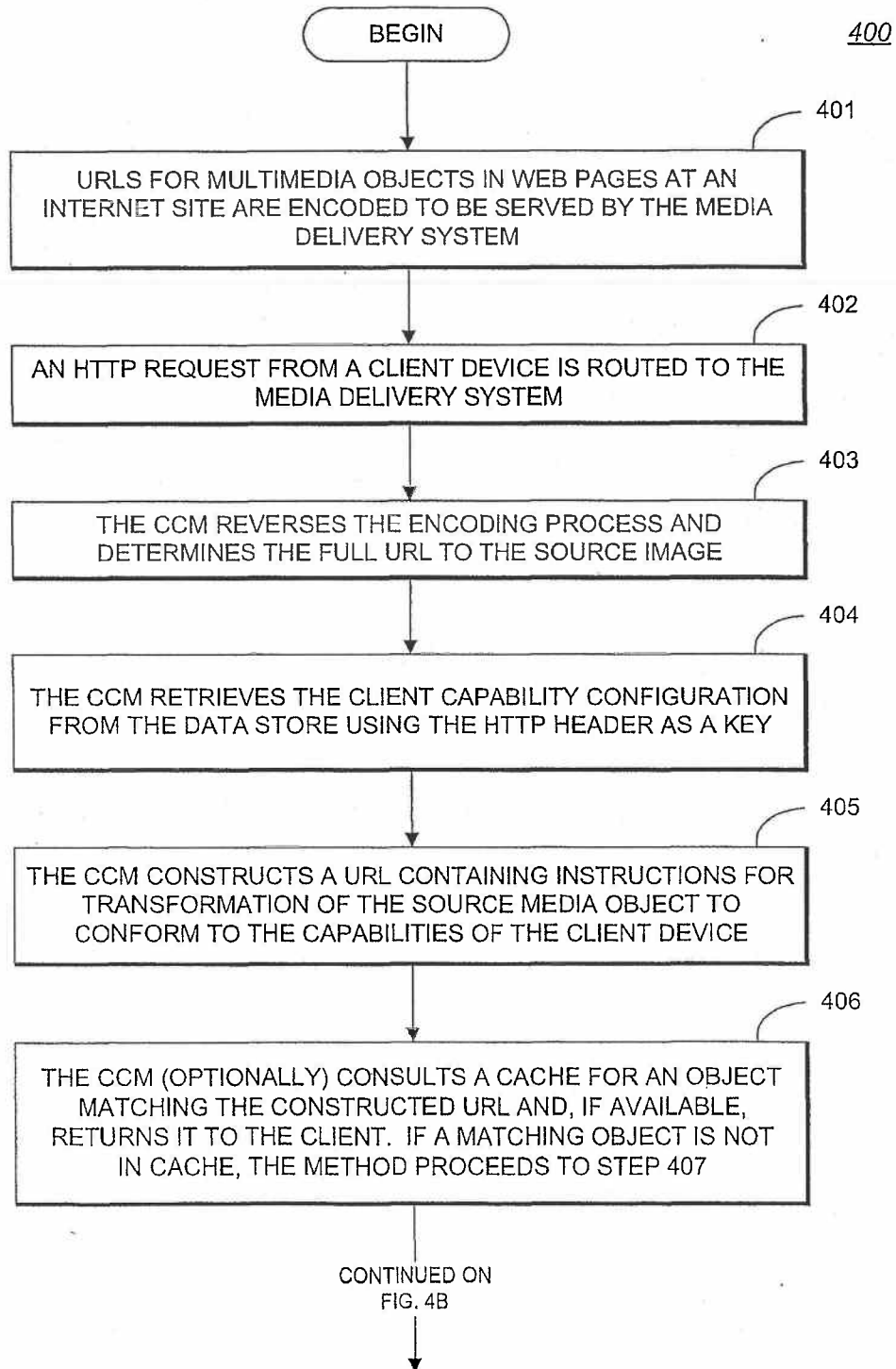
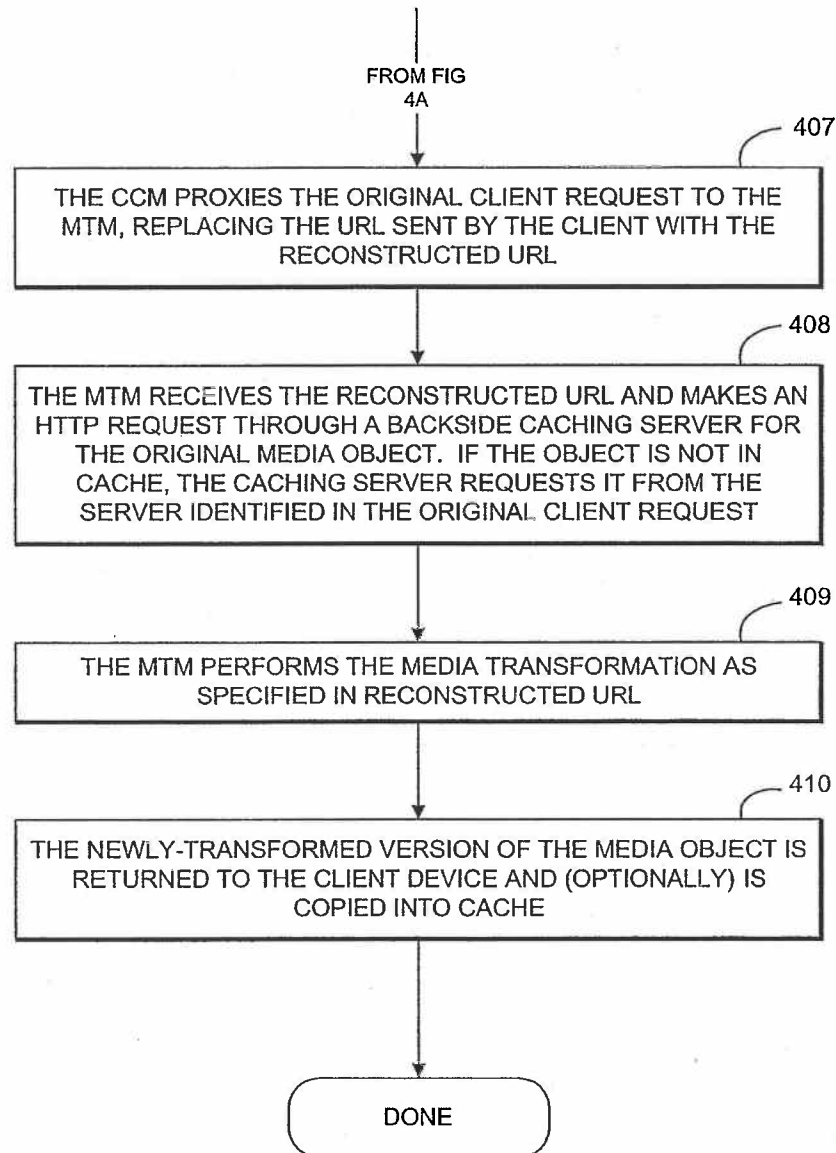


FIG. 4A

**FIG. 4B**

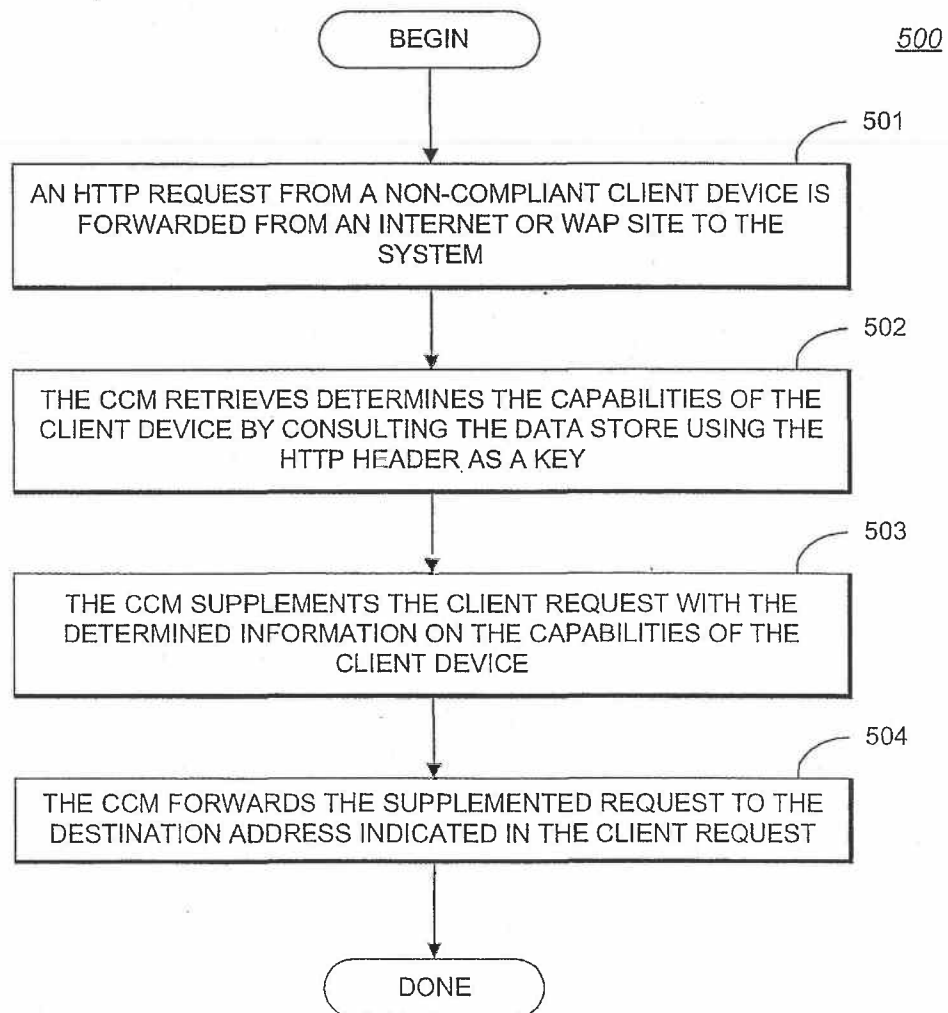


FIG. 5

APPENDIX 24

ONA: DEMANDS ON PROVISIONING AND PERFORMANCE

S. Homayoon,

G. Giridharagopal

Northern Telecom Inc.
RTP, NC

BNR Inc., RTP, NC

Abstract

ONA promises to stimulate and make information services available to the general public. These services represent a market in excess of \$10B in 1988, growing to \$30B in 1995.

Key to making the promise of ONA a reality is the increased use of the public switched network at a reasonable cost.

The current industry debate surrounding the approval and implementation of the initial phases of ONA centers on issues of unbundling, packaging and pricing of the capabilities of the public network. This paper focuses on a few critical technical areas - real time capacity provisioning and performance - where ONA is expected to impact the public network.

1. Introduction

The evolution of the public network to support enhanced services creates the need for planning the growth of real-time switch capacity in concert with the emergence of these new services. A methodology for provisioning the real-time capacity of a digital switch is described in Section 2 and examples of its application to assess the real-time processing requirements of some future services are presented. Performance aspects of the future intelligent networks and the increase in processing required in networks with distributed intelligence is addressed in Section 3.

2. Real-time Provisioning Methodology

Services supported by future networks will require additional software and more real-time processing than the traditional Plain Ordinary Telephone

Services (POTS). In addition, ONA requires existing BOC services to be unbundled into elemental service capabilities, called 'Basic Service Elements (BSE)'. The unbundling of services will also increase the real-time processing requirements. It is therefore necessary to estimate the real-time requirements to facilitate the planning of growth in real-time demand.

The average real-time requirement per subscriber line consists of two components: (a) real-time required due to feature assignments and (b) real-time required due to feature activation or usage.

Average real-time required per line due to feature penetration

$$= \sum_i N_i \cdot \sum_m f_{im} \cdot S_{im} \quad (1)$$

where, N_i = fraction of lines belonging to customer type i

f_{im} = fraction of type i lines that have feature m

S_{im} = real-time due to assignment of feature m to customer type i

Average real-time required per line due to feature activation (attempt)

$$= \frac{\sum_i A_i \cdot N_i \cdot \sum_n t_{ni} \cdot \sum_k p_k}{\sum_m f_{im} \cdot a_{im} \cdot g_{inkm}} \quad (2)$$

where, A_i = total HDBH attempts per line of type i

t_{ni} = fraction of attempts from type i lines that belong to call type n

g_{inkm} = real-time required per attempt involving feature m on line type i , call type n , meeting call disposition type k

32.5.1.

a_{im} = fraction of attempts from type i lines that invoke feature m
 p_k = probability of a call meeting disposition type k .

Average real-time required per line

$$= \frac{\sum_i N_i \cdot \sum_m f_{im} \cdot [s_{im} + A_i \cdot \sum_n \sum_k t_{nk} \cdot p_k \cdot a_{im} \cdot g_{inkm}]}{\sum_k t_{nk} \cdot p_k \cdot a_{im} \cdot g_{inkm}} \quad (3)$$

The number of lines that can be supported can be computed from (3), given the CPU occupancy available for call processing (C) as

$$\text{Number of lines} = (C \cdot 3600 \cdot 1000) / (\text{Average Real-time per Line in msec})$$

2.1 Real-time Requirements of Future Services

To estimate the real-time processing demand placed on a digital switch by future services, two services are analyzed below: **voice messaging** and **pay-per-view television**. These services are currently under consideration as being attractive to Enhanced Service Providers (ESPs) under the ONA framework, based on the Basic Service Elements (BSEs) provided through the public network.

2.1.1 Voice Messaging

Voice messaging service, illustrated in Figure 1, enables an ESP to receive messages destined for its clients, when the client's line is forwarded to the ESP. The ESP stores the message and sends a message waiting indication to the client through the public network. On receiving the message waiting indication, the client can call the ESP to retrieve the message(s), at which time the ESP sends a message through the public network requesting turn-off of the message waiting indication at the client's line.

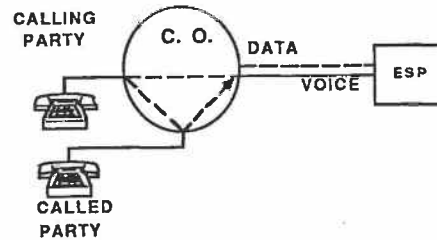


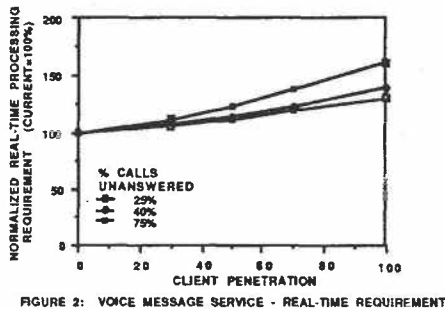
Figure 1: Voice Messaging

The following BSEs are required to enable the provision of the voice messaging service [1]

- call forwarding feature assigned to client's line (busy or don't answer)
- distribution of calls over several message desk attendants
- delivery of the calling customer's identification
- delivery of the identity of the ESP's client who is called
- identification of the reason the call was forwarded
- identification of the message desk port to which the call is delivered
- message waiting indication on/off.

These BSEs are all supported today by the Simplified Message Desk Interface (SMIDI) [2]. The capability is currently available only on an intra-switch basis. The penetration of the service and fraction of customer calls that are forwarded to the ESP are two key parameters influencing the real-time processing requirements of voice messaging. Typically, 40% of a business customer's calls are unanswered. Using equations (1) and (2), and a 40% activation rate for voice messaging service indicates that a voice message call will require about 4 to 5 times as much real-time as a simple intra-office POTS call. As illustrated in Figure 2, at the anticipated penetration of 30%, assuming that 40% of customer calls are routed to the ESP, the voice messaging service will result in a 15% increase in the real-time demand placed on a digital switch. If the usage were to increase to 75%, the real-time requirement for voice messaging will increase marginally to about 22% at the same penetration.

32.5.2.



2.1.2 Pay-per-view Television

Pay-per-view television service, illustrated in Figure 3, can be provided to cable television subscribers who have remotely addressable cable decoders. The service works as follows: A customer dials a specific number provided by the cable company as identifying the program that the customer wishes to watch. The customer also tunes the television to the specific channel and receives a scrambled signal. The customer's call is routed through the BOC network.

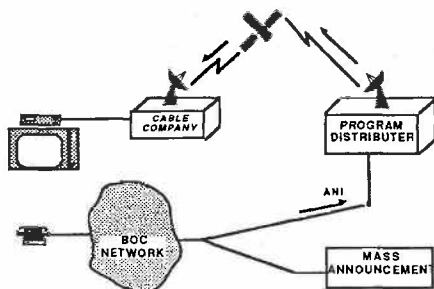


FIGURE 3: PAY PER VIEW TELEVISION

The BOC delivers the subscriber's identity (in the form of ANI) to the ESP. The ESP or the BOC plays an announcement to indicate to the subscriber that the order is being processed. The ESP uses the customer's identity to validate the request and send a signal to the remote decoder to decode the signal on the specific channel. As confirmation of the order, the customer sees a clear, descrambled signal. The pay-per-view television service applies mainly to residential customers.

This enhanced service requires one BSE: delivery of ANI. The BOC may also provide Central Office Announcements as a BSE, if requested by an ESP. Service penetration and service usage are the key parameters which determine the real-time impact of this service similar to the voice messaging-service. Figure 4 shows the increase in switch real-time processing requirements as a function of these two parameters. Assuming a service penetration of 30% and based on a busy hour attempt rate of one per customer, the pay-per-view service will increase the real-time demand on the switch by about 11 percent.

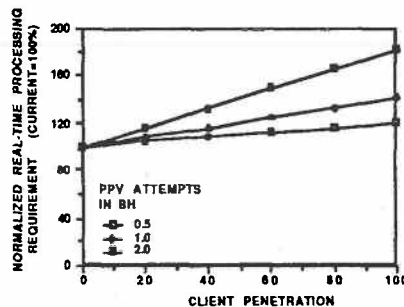


Figure 4: Pay Per View Service - Real-time Requirement

ONA will accelerate the penetration of enhanced services and the BOCs will get additional revenues from the increased network usage. However, it is necessary to plan the growth of switch real-time capacity to support these enhanced services.

3. Future Network Performance

The network of today has evolved based on intelligence, such as features and call processing software, located primarily at the local central office switches and accessed by relatively unintelligent terminals. New enhanced services being considered are likely to need network capabilities typically supported at some centrally located 'intelligent network elements'. With the emergence of SS7, the class-5 office can access these capabilities. Some of the intelligence traditionally provided at the class-5 office is also moving to the customer terminals; primary examples being ISDN and personal computers. Thus, the trend in future networks will be to

provide additional intelligence in the Customer Premises Equipment (CPE) or at intelligent Network Elements (NE)

[3]. The degree to which the intelligence is distributed will depend on feasibility and cost and the requirements imposed by potential new services. The distribution of intelligence will impact the call set-up times in future networks. An example is considered in Section 3.1 to quantify this impact.

The network will also be required to handle large volumes of different types of traffic, ranging from the traditional POTS to the complex multi-media services. Fiber optic transmission become attractive as the technology of choice to support the large bandwidth requirements of future networks. The heavy concentration of traffic onto a few large transmission paths raises concerns about possible degradation in network performance resulting from reduced network survivability [4,5]. These concerns have risen with every step in technology, such as the introduction of PCM systems, but with fiber they become more important because of the very low connectivity that may result from the high capacity of fiber systems. Thus, the emergence of fiber transport systems has intensified the need to develop methods for improving the survivability of future networks so that current customer expectations can be maintained. At BNR, a method for designing robust transport networks has been developed [6].

3.1 Call Set-up Time

The distribution of intelligence in the emerging networks, as illustrated in Figure 5, will require more resources to maintain the call set-up time at today's levels. To compensate for this, it will be necessary to identify the optimum positioning of network intelligence and possibly consider higher speed signaling links (e.g., increase from the current 64 kbps to say 1.5 mbps).

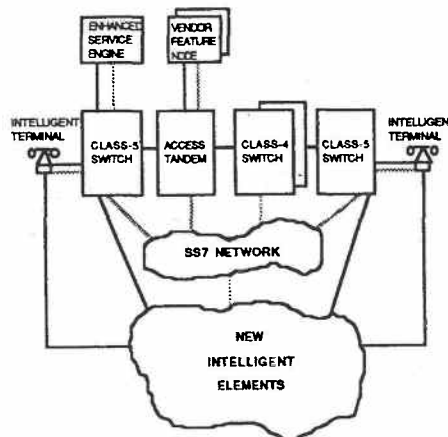


FIGURE 5: THE FUTURE INTELLIGENT NETWORK

Consider the case of a call requiring interaction with the ESP's equipment. There are two time factors in this call: (1) Call Set-up Time: the time from the receipt of the last dialed digit at the originating office to the time an announcement is played by the ESP's node (2) Cut Through Time: the time from the completion of the dialling procedure (based on instructions from the Enhanced Services Engine (ESE)) to the time a 'welcome' message is played by the ESP's equipment. There are two possible cases of intelligence distribution: (1) all intelligence is supported at the originating switch and (2) all intelligence is supported at other network elements, requiring temporary "hand-off" of call processing between the originating switch and other intelligent network elements. The call set-up time will increase with the complexity of the hand-off mechanism.

A simple analysis was performed to estimate the increase in call set-up times. The analysis was based on the assumptions of a M/M/1 queue at each node with server occupancies varying between 40% (CCS7 links) to 70% (digital switches). Information packets were assumed to be of 200 octets and supervisory packets had a size of 50 octets. Call anomalies and their associated recovery procedures were not considered. Signaling between network elements was assumed to be based on CCS7.

32.5.4.

Figure 6 summarizes the call set-up times in these two cases. Average call set-up time increases by almost 200%, to about 1.2 seconds, with distribution of network intelligence away from the originating switch. This is smaller than the call set-up times experienced in the current network based on MF signaling. However, it is larger than the call set-up time that can be expected in the current network if CCS7 were used for inter-office signaling [6].

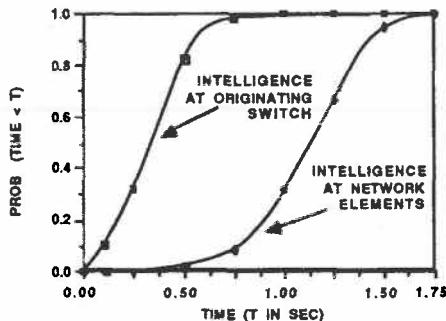


FIGURE 6: CALL SET-UP TIMES

4. Conclusions

Services to be supported in the future intelligent networks will require more software and real-time processing than existing services. Open Network Architecture will accelerate the penetration of these enhanced services and will require that access to existing switch call processing software be provided at an elemental service level. The real-time demand on SPC switches will increase, and planning for graceful evolution in real-time switch capacity will become necessary.

The intelligent networks of the 1990's are likely to have network intelligence distributed among various network elements, instead of being concentrated at the local switch. More resources will be needed to maintain call set-up times at current levels. Performance grades of service must be defined based more on service classifications than on network elements, so that appropriate locations for intelligence can be determined to meet the service performance criteria.

References

- [1] ONA Forum2, Los Angeles, CA., USA, Presentation by Southwestern Bell Telephone Company, March 30, 1987.
- [2] Bell Communications Research, "Interface Description - Interface Between Customer Premise Equipment, Simplified Message Desk and Switching System: 1AESS", TR-TSY-000283, Issue 1, July 1985.
- [3] Bell Communications Research, "Plan for the Second Generation of the Intelligent Network", Special Report, SR-NPL-000444, Issue 1, July 1986.
- [4] Kaiser, P., Midwinter, J. and Shimada, S., "Status and Future Trends in the Terrestrial Optical Fiber Systems in North America, Europe and Japan", IEEE Communications, October 1987.
- [5] Gruber, J.G., et al, "Quality of Service in Telecommunications Networks", IEEE Journal on Selected Areas in Communications, Vol. SAC-4, No. 7, October 1986.
- [6] Sabat, J., "An Algorithm for Improving Survivability in a Fiber Based Network", Fifth International Workshop on Integrated Electronics and Photonics in Communications, October 21-23, 1987, Raleigh, NC. USA (Sponsored by IEEE).
- [7] Horing, S., et al, "Stored Program Controlled Network: Overview", The Bell System Technical Journal, Vol. 61, September 1982.

APPENDIX 25

Expert Qualifications - Michael Perkins Ph.D.

1. I received Ph.D. and MS degrees from Stanford University in Electrical Engineering, an MS degree from Stanford University in Statistics, and a BS degree from the University of Kansas in Electrical Engineering. I have worked in the area of image and video compression, transmission, storage, and display since 1988 as a research scientist, engineer, and engineering manager.

2. I co-founded three companies, DiviCom, Vantum, and Cardinal Peak, all of which specialize in video compression, processing, and transmission. While employed by Scientific Atlanta, one of the world's leading manufacturers of settop boxes, I served as Scientific Atlanta's principal representative to the MPEG (Moving Pictures Expert Group) standards committee (1992 – 1993), and later as DiviCom's principal representative (1993-1995). I was involved in the creation of the MPEG-2 standard, including drafting various sections related to systems issues.

3. I have 20+ years of experience designing, developing, and managing the development of a variety of real-time systems for video and data processing. At the time Patent Number '592 was filed, I worked at Cardinal Peak and was providing consulting services to a variety of firms. I was responsible for the development of DiviCom's MPEG encoding algorithms and data over cable products. At Vantum, I was Vice President of Engineering and led the development of their award-winning video over IP camera, which streamed compressed MPEG video over IP networks in addition to storing the video to a hard drive. Presently I am a managing partner at Cardinal Peak, LLC, and have been responsible for the development of a digital video recording and playback system for law enforcement applications, including police cars and interview rooms. In addition to my current position at Cardinal Peak, I have taught as an Adjunct Professor in the Engineering Department at the Colorado School of Mines.

4. I am a named inventor on 7 patents, have published a variety of articles in peer-reviewed professional journals and conference proceedings, and have made numerous technical presentations.

Mike Perkins, Ph.D.

SUMMARY Twenty five years of experience in research, algorithm development, system design, engineering management, executive management, and Board of Directors membership. Also skilled in technical sales, customer relations, writing, and presentations. Possess valuable small business experience through the founding of three companies.

PROFESSIONAL EXPERIENCE **Cardinal Peak, LLC, Boulder, Colo.** Feb 2002 to Present
(co-founder)

Managing Partner: Responsibilities include software development, technical consulting, management of contract engineering projects, business development, and management of various aspects of day-to-day operations.

Colorado School of Mines, Golden, Colo. Aug 2002 to 2007

Adjunct Professor: Teach classes for the engineering department as needed in the areas of communications, linear systems, and signal processing.

Vantum Corporation, Boulder, Colo. Sept 1999 to Feb 2002
(co-founder)

VP Engineering: Built and led engineering team responsible for the development of Vantum's products. Products included a programmable video appliance for compressing, storing, and streaming MPEG video over IP networks; a software development tool for writing JavaScript programs for the video appliance; various software utilities to support deployed networks of video appliances.

DiviCom, Inc., Milpitas, Calif. May 1993 to Sept 1999
(co-founder)

Engineering Director, Headend Data Systems: Built and led engineering team responsible for the development of DiviCom's data systems products. Products comprised a headend data termination system (Ethernet interface, IP routing, QPSK modulation and demodulation, broadband MAC protocol); an Ethernet / IP to MPEG-2 transport stream encapsulator and router; and a general purpose PC-based MPEG-2 manipulation toolkit.

Engineering Director, Video Quality and Algorithm Development: Built and led engineering team responsible for developing video processing and encoding algorithms, including rate control, statistical multiplexing, ATM traffic shaping, and network de-jittering. Responsibilities included managing the development of the encoder boards in DiviCom's mainstream encoder products. Served as DiviCom's principal representative to the MPEG-2 committee and to the ATM Forum for 1½ years, authoring numerous contributions.

Scientific Atlanta, Inc., Norcross, Georgia Aug 1991 to May 1993

Staff Engineer: Responsible for SA's efforts in the area of video compression algorithm development. Responsibilities included: serving as SA's principal delegate to the MPEG committee; simulations of MPEG and VQ video encoding algorithms; collaboration and liaison with the Utah State University team contracted to develop a real-time VQ encoder; frequent presentations to top executives of SA and its customers on compression technologies and strategies.

The German Aerospace Research Establishment: Institute for Communications Technology, Department of Communications Theory, Oberpfaffenhofen, Germany

Oct 1988 to Aug 1991

Research Scientist: Performed research in the area of combined source-channel coding. Techniques investigated include joint quantizer/signal-constellation design, DCT image coding coupled with various error-control techniques, combined tree-structured quantization and channel coding, soft-

Mike Perkins, Ph.D.

decision decoding of block codes, and channel coding for source coders that use Huffman coding.

Stanford University, Stanford, Calif. *1983 to 1988*

Post-Doctoral Fellow: Subband coding of images using psychophysically justified bit allocations.

Research Assistant: Provided analysis and simulation support for the development of an A/D converter that flew as part of a space shuttle experiment. Performed thesis research in the area of data compression of stereopairs.

Hughes Aircraft Corporation, El Segundo, Calif. *Summer 1983 and 1984*

Member of Technical Staff: Simulated FDMA communication systems. Modified and verified the accuracy of test procedures for the space shuttle Ku-band communication system; automated the test procedures through software control of the instrumentation.

EDUCATION

Stanford University, Stanford, Calif. *1983 to 1988*

Ph.D., Electrical Engineering, March 1988

M.S., Statistics, December 1986

M.S., Electrical Engineering, June 1984

University of Kansas, Lawrence, Kansas *1979 to 1983*

B.S., Electrical Engineering, May 1983

PATENTS

Granted

- | | |
|-----------|---|
| 5,420,639 | M. Perkins, "Rate-adaptive Huffman coding" |
| 5,717,464 | M. Perkins, D. Arnstein, "Rate control for a video encoder" |
| 5,920,572 | E. Washington, M. Perkins, et. al., "Transport stream decoder/demultiplexer for hierarchically organized audio-video Streams" |
| 5,828,414 | M. Perkins, T. Lookabaugh, "Reduction of timing jitter in audio-video transport streams" |
| 5,859,660 | M. Perkins, W. Helms, "Non-seamless splicing of audio-video transport streams" |
| 5,861,919 | M. Perkins, D. Arnstein, "Dynamic Rate Optimization for an Ensemble of Video Encoders" |
| 6,188,729 | M. Perkins, "Method and apparatus for effecting seamless data rate changes in a video compression system" |

PUBLICATIONS

Journals

- M. Perkins and D. Arnstein, "Statistical Multiplexing of Multiple MPEG-2 Video Programs in a Single Channel," *SMPTE Journal*, Vol. 104, No. 9, September 1995
- M. Perkins, "Data Compression of Stereopairs," *IEEE Transactions on Communications*, Vol. 40, No. 4, April 1992
- T. Lookabaugh and M. Perkins, "Application of the Princen-Bradley Filter Bank to Speech and Image Compression," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, Vol 38, No. 11, Nov. 1990
- M. Perkins, "A Comparison of the Hartley, Cas-Cas, Fourier, and Discrete Cosine Transforms for Image Coding," *IEEE Transactions on Communications*, Vol. 36, No. 6, June, 1988
- M. Perkins, "A Separable Hartley-Like Transform in Two or More Dimensions," *Proceedings of the IEEE*, Vol. 75, No. 8, August, 1987

Conference Proceedings

- J. Zhang and M. Perkins, "Effects of ATM Transmission Errors on MPEG-2 Quality of Service," SMPTE Conference, 1995?

Mike Perkins, Ph.D.

- M. Perkins and E. Offer, "Combined Source-Channel Image Coding Using Tree-Structured Vector Quantization and Sequential Decoding," International Conference on Communications, (ICC), June, 1991
- E. Offer and M. Perkins, "Soft-Decision Decoding of Block Codes Using the Stack Algorithm," International Conference on Communications (ICC), June, 1991
- M. Perkins and T. Lookabaugh, "Combined Source-Channel DCT Image Coding for the Gaussian Channel," Proceedings of the European Signal Processing Conference (Eusipco), September, 1990
- T. Woerz and M. Perkins, "Improving the Performance of a Low-Rate Image Coder Connected to a Noisy Gaussian Channel," Proceedings of the European Signal Processing Conference (Eusipco), September, 1990
- M. Perkins, "Joint Vector Quantization and Signal Constellation Design for the Gaussian Channel," International Symposium on Information Theory, January, 1990
- M. Perkins, "Optimizing Signal Constellations for the Output of a Quantizer," Proceedings of IEEE Global Telecommunications Conference (Globecom), November, 1989
- M. Perkins and T. Lookabaugh, "A Psychophysically Justified Bit Allocation Algorithm for Subband Image Coding Systems," Proceedings of International Conference on Acoustics Speech and Signal Processing (ICASSP), May, 1989
- T. Lookabaugh, M. Perkins, and C. Cadwell, "Analysis/Synthesis Systems in the Presence of Quantization," Proceedings of International Conference on Acoustics Speech and Signal Processing (ICASSP), May, 1989

Miscellaneous

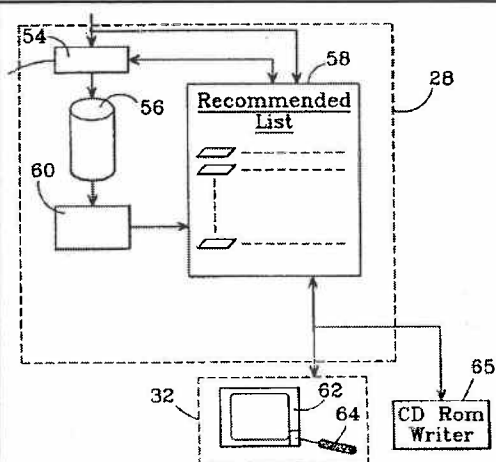
- M. Perkins, "Unlocking the True Broadband Potential of Cable Networks," *IBC Daily News*, Sept. 12, 1998
- M. Perkins, "Digital Video and Data," *International Cable*, May, 1998
- M. Perkins, "Data Compression of Stereopairs," Stanford University Ph.D Dissertation, February, 1988

APPENDIX 26

CONFIDENTIAL - ATTORNEY WORK PRODUCT
Claim Chart for U.S. Patent No. 7,505,592 in view of Payton (U.S. Patent 5,790,935)

Claim 1	Sections of Payton
<p>[A] A method of transmitting service instances to a remote client device in a local network, including a communications device and one or more of said remote client devices, at a subscriber premises, comprising the steps of:</p>	<div data-bbox="641 346 1356 808" data-label="Diagram"> <p style="text-align: center;">FIG. 2</p> </div> <p>“The present invention provides virtual on-demand delivery of digital information over existing, as well as next generation, digital transport systems.” 4:7-9.</p> <p>Service Instance: “The digital items may include videos, audio selections and computer applications.” 4:57-58.</p> <p>Remote Client Device: “In response to a subscriber’s request, the delivery system 22 delivers the requested item to the subscriber’s playback device 32 such as a television, audio system or computer.” 4:52-54; “The subscriber’s playback device 32 preferably includes some type of a video display 62 such as a T.V. or a computer monitor and a control device 64 such as a remote or a keyboard.” 6:20-25.</p> <p>In a local network at a subscriber premises: The local network includes the local server 28 and the playback device 32. <i>See Fig. 2.</i></p> <p>Communications Device: “local server 28 for each subscriber” 4:47-48.</p>
<p>[B] receiving a service instance at the</p>	<p>Receiving a service instance at the communications device of the subscriber premises: The digital transport</p>

<p>communications device of the subscriber premises from a headend, wherein the service instance is stored in memory of the communications device;</p>	<p>system 26 can be a satellite or cable network and can include “a central transmitter 48, a broadcast satellite 50, and a plurality of local receivers 52” for distributing digital items 36 from the central transmitter 48 to the local receivers 52 via the broadcast satellite 50. 5:59-67.</p> <p>From a headend: “central distribution server 24” 4:46.</p> <p>“The central distribution server 24 includes a digital repository 34 for storing all of the digital items 36 that will be made available to the subscribers.” 4:55-57.</p> <p>“The digital transport system 26 broadcasts the digital items 36 from the central distribution server 24 to each of the local servers 28.” 5:55-57.</p> <p>Wherein the service instance is stored in memory of the communications device: “Each subscriber has a local server [(28)] that downloads recommended and specifically requested items and stores them in a local storage device [(56)].” 2:67-3:2.</p> <p>“As a requested item 36 is being received, it is downloaded to the local storage 56.” 6:11-12.</p>
<p>[C] receiving a request at the communications device for the stored service instance from the remote client device;</p>	<p>“To request an item, the subscriber interface 58 displays the list 44 of recommended items...on the video display. The subscriber can either select one of the displayed items using the control device [(64)] or request a menu of the remaining available items. Alternatively, the initial menu might show all offerings with the recommended items highlighted.” 6:26-33.</p> <p>“As shown in FIG. 3b, when a subscriber makes an on-demand request (step 74), the local server 28 polls the local storage 56 (step 75) to determine whether the requested item is stored locally (step 76). 7:13-16, Figs. 2, 3b:</p> <div data-bbox="987 1482 1409 1650" data-label="Diagram"> <pre> graph TD 74[Subscriber's On-Demand Request] --> 75[Poll Local Storage] </pre> </div>



"The back channel 30 is used ... to transmit the subscribers' requests ... from the subscribers back to the central distribution server 24." 6:51-53.

[D] determining whether the remote client device is entitled to receive the requested stored service instance; and

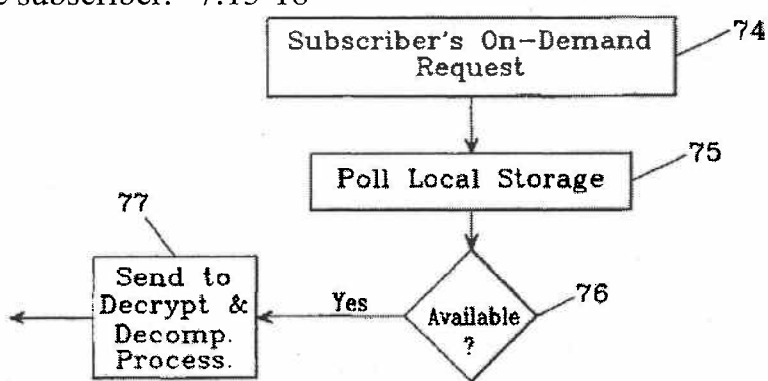
There are several alternative features within Payton that meet this limitation. *See, e.g.:*

- "Furthermore, the digital items are preferably encrypted so that downloaded items cannot be accessed without first being paid for." 4:64-66. Inherently, at some point, a determination is made whether the playback device can access a service.
- The distribution server may limit what selections are available:
 - The local server 28 contacts the distribution server 24 regarding profile and billing data. 7:65-8:10. A determination of entitlement is inherent in determining what items the distribution server will permit for distribution.
 - "If the system becomes loaded, the scheduling processor 46 limits the number of items, in addition to those stored on the subscribers' local servers, that are available to the subscribers. This has the effect of reducing the number of on-demand requests made to the central distribution server." 5:40-45. The distribution server can broadcast a message to subscribers to reduce menu selections in order to reduce the load on the transport system. 7:25-35. If the menu selection is disabled or removed, the subscriber is consequently not entitled to that selection.

	<ul style="list-style-type: none"> ○ “Furthermore, the local subscriber’s requests can be limited to those on their recommended lists. 5:49-51. ○ Alternatively, in one embodiment, a subscriber may be limited to selections stored on the local video server and not entitled to a video from the central distribution server. 9:62-10:9.
[E] responsive to determining the remote client device is entitled to receive the requested stored service instance, transmitting the requested service instance from the communications device to the remote client device.	<p>“In response to a subscriber’s request, the delivery system 22 delivers the requested item to the subscriber’s playback device 32.” 4:52-54</p> <p>“As a requested item 36 is being received, it is downloaded to the local storage 56, while a decryption and decompression processor 60 decrypts, decompresses and converts the item from the digital format into a standard video or audio signal, and sends the signal to the interface control which routes the signal to the subscriber’s playback device 32.” 6:11-17</p>
Claim 2	Sections of Payton
[A] The method of claim 1, further comprising the steps of: determining a device type for the remote client device;	<p>A device type is inherently determined in the manner of distribution of material to either the CD Rom writer or the TV. The service instance can be “routed to the playback device 32 or CD ROM writer 65.” 8:19-20. At some point, the system would determine that it is communicating data in the direction of a CD ROM writer as opposed to a TV.</p>
[B] using the device type of the remote client device to determine a format protocol	<p>“the item is transferred to the decryption and decompression processor...and then routed to the playback device 32 or CD ROM writer 65.” 8:18-20.</p>

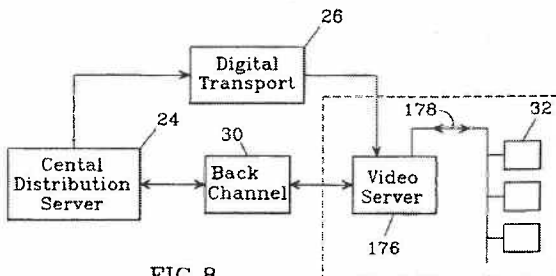
for the requested service instance; and	Inherently, a format protocol would be determined before directing data towards the CD ROM writer or the TV.
[C] reformatting the requested service instance prior to transmitting the requested service instance.	...and inherently, the service instance would be reformatted prior to transmission to the CD ROM writer or TV. The TV and the CD ROM writer use different signal formats.
Claim 3	Sections of Payton
[A] The method of claim 1, further comprising the steps of: determining an encryption scheme for the requested service instance based on the remote client device; and	
[B] prior to transmitting the requested service instance, encrypting the requested service instance.	
Claim 4	Sections of Payton
[A] The method of claim 3, further comprising the steps of: receiving a message along with the request from the remote client device, the message having device type information regarding the remote client device; and	
[B] determining from the device type information the encryption scheme for encryption of the requested service instance.	
Claim 5	Sections of Payton
[A] The method of claim 3, further comprising the steps	

of: receiving a message along with the request from the remote client device, the message having device type information regarding the remote client device;	
[B] transmitting a second message to the headend, the second message having the device type information included therein, wherein the headend determines the encryption scheme using at least the device type information.	
Claim 6	Sections of Payton
[A] A method of transmitting service instances to a remote client device in a local network, including a communications device and one or more of said remote client devices, at a subscriber premises, comprising the steps of:	(See Claim 1 - element [A]).
[B] receiving a request at the communications device of the subscriber premises for a service instance from the remote client device;	See Claim 1 - element [C].
[C] determining whether the remote client device is entitled to receive the requested service instance;	See Claim 1 - element [D].
[D] responsive to determining the remote	"In response to a subscriber request, the local server first polls the local storage to see if the requested item is stored

<p>client device is entitled to receive the requested service instance, determining whether the requested service instance is accessible within the communications device;</p>	<p>locally.” 3:25-27.</p>
<p>[E] if the requested service instance is accessible, transmitting the service instance from the communications device to the remote client device; and</p>	<p>“As shown in FIG. 3b, when a subscriber makes an on-demand request (step 74), the local server 28 polls the local storage 56 (step 75) to determine whether the requested item is stored locally (step 76). If it is stored locally, the item is sent to the decryption and decompression processor (step 77) 60 and then routed to the subscriber.” 7:13-18</p>  <pre> graph TD 74[Subscriber's On-Demand Request] --> 75[Poll Local Storage] 75 --> 76{Available?} 76 -- Yes --> 77[Send to Decrypt & Decomp. Process.] 77 --> Exit(()) </pre>
<p>[F] if the requested service instance is not accessible, sending a request from the communications device to a headend for the requested service instance, wherein upon receipt of the service instance at the communications device, transmitting the service instance from the communications device to the remote client device.</p>	<p>“In response to a subscriber request, the local server first polls the local storage to see if the requested item is stored locally. If the item is not immediately available, the local server sends the request over the back channel to the central distribution server, which places the request in the queue for on-demand broadcast. As soon as the item is received locally, it is sent to a local playback device.” 3:25-33.</p> <p>“As shown in FIG. 3b, when a subscriber makes an on-demand request (step 74), the local server 28 polls the local storage 56 (step 75) to determine whether the requested item is stored locally (step 76). If it is stored locally, the item is sent to the decryption and decompression processor (step 77) 60 and then routed to the subscriber. Otherwise, the local server polls the central distribution processor by sending the request over the</p>

	<p>back channel (step 78).” 7:13-20</p> <pre> graph TD 75[75 Poll Local Storage] --> 76{76 Available?} 76 -- Yes --> 77[77 Send to Decrypt & Decomp. Process.] 76 -- No --> 7[7 Poll Central Distribution Processor] 7 --> 78[78 Receive On-Demand Request] 78 --> 80{80 In Queue?} 80 -- Yes --> 77 80 -- No --> 78 </pre>
Claim 7	Sections of Payton
<p>The method of claim 6, further comprising the step of, prior to transmitting the service instance to the remote client device, storing the received service instance in memory of the communications device.</p>	<p>6:11-17 (See Claim 1 - element [E]).</p> <p>“As shown in FIG. 5, to download broadcast items the prediction filter 54 receives the broadcast item (step 124) from the local receiver 52 or from broadcast television. The filter then determines whether the item is an on-demand request from the subscriber (step 126). If it is, the filter determines whether the pause control is on (step 128), and if so routes the item to the local storage 56 (step 130). Once the pause control is released, the item is transferred to the decryption and decompression processor 60 (step 132) and then routed to the playback device 32... The item is simultaneously transferred to local storage to allow the user to rewind at any time. Also, if the pause control is released during transfer to storage, the item is transferred to decryption from storage while storage continues to be loaded from the received broadcast.” 8:11-25.</p>
Claim 8	Sections of Payton
<p>[A] The method of claim 6, further comprising the steps of: determining a device type for the remote client device;</p>	<p>See Claim 2 - element [A].</p>
<p>[B] using the device type of the remote client device to</p>	<p>See Claim 2 - element [B].</p>

determine a format protocol for the requested service instance; and	
[C] reformatting the requested service instance prior to transmitting the requested service instance.	...and inherently, the service instance would be reformatted prior to transmission to the CD ROM writer or TV. The TV and the CD ROM writer use different signal formats.
Claim 9	Sections of Payton
[A] The method of claim 6, further comprising the steps of: determining an encryption scheme for the requested service instance based on the remote client device; and	
[B] prior to transmitting the requested service instance, encrypting the requested service instance.	
Claim 10	Sections of Payton
[A] The method of claim 9, further comprising the steps of: receiving a message along with the request from the remote client device, the message having device type information regarding the remote client device; and	
[B] determining from the device type information the encryption scheme for encryption of the requested service instance.	
Claim 11	Sections of Payton
[A] The method of claim 9,	

<p>further comprising the steps of: receiving a message along with the request from the remote client device, the message having device type information regarding the remote client device;</p>	
<p>[B] transmitting a second message to the headend, the second message having the device type information included therein, wherein the headend determines the encryption scheme using at least the device type information.</p>	
Claim 12	Sections of Payton
<p>[A] A method of transmitting service instances to a plurality of remote client devices in a local network, including a communications device, at a subscriber premises, comprising the steps of:</p>	<p>Fig. 2 (See Claim 1 - element [A]). 4:7-9 (See Claim 1 - element [A]). 4:57-58 (See Claim 1 - element [A]). <i>Plurality of remote client devices:</i> 6:20-25 (See Claim 1 - element [A]).</p> <p><i>Regarding the plurality of remote client devices, Payton teaches a plurality of local subscribers connected to a video server 176. 9:62-10:20; Fig. 8.</i></p>  <p style="text-align: center;">FIG. 8</p> <p>In a local network at a subscriber premises: The local network includes the local server 28 and the playback device 32. See Fig. 2. <i>Payton teaches the plurality of local subscribers 32 in a network (176;178;32) in Fig. 8.</i></p>

	<p>Communications Device: "local server 28 for each subscriber" 4:47-48. <i>For the embodiment with the plurality of remote client devices, the communications device is the video server 176.</i></p>
<p>[B] receiving a first request for a service instance at the communications device of the subscriber premises from a first remote client device;</p>	<p>See Claim 1 - element [C].</p> <p><i>Regarding the requirement that this be a first request from a first remote client device, Payton teaches a plurality of local subscribers connected to a video server 176. 9:62-10:20; Fig. 8. These constitute multiple subscribers making multiple requests from multiple client devices. In this embodiment, "each local server has been replaced by a video server 176 which serves a plurality of local subscribers." 9:63-64. The system operates as it otherwise would with a single subscriber.</i></p>
<p>[C] determining whether the first remote client device is entitled to receive the requested service instance;</p>	<p>See Claim 1 - element [D].</p>
<p>[D] responsive to determining the first remote client device is entitled to receive the requested service instance, transmitting the requested service instance from the communications device to the first remote client device;</p>	<p>See Claim 6 - element [E].</p>
<p>[E] receiving a second request for the service instance at the communications device from a second remote client device in the local network;</p>	<p>See Claim 1 - element [C].</p> <p><i>Regarding the requirement that this be a second request from a second remote client device, Payton teaches a plurality of local subscribers connected to a video server 176. 9:62-10:20; Fig. 8. These constitute multiple subscribers making multiple requests from multiple client devices. In this embodiment, "each local server has been</i></p>

	<i>replaced by a video server 176 which serves a plurality of local subscribers.” 9:63-64. The system operates as it otherwise would with a single subscriber.</i>
[F] determining whether the second remote client device is entitled to receive the requested service instance; and	See Claim 1 - element [D].
[G] responsive to determining the second remote client device is entitled to receive the requested service instance, transmitting the requested service instance from the communications device to the second remote client device based upon local availability of the requested service instance within the communications device.	See Claim 6 - element [E]. The limitation of “local availability” is anticipated by the above polling for availability and subsequent distribution upon confirmation that the requested instance is available.
Claim 13	Sections of Payton
[A] The method of claim 12, further comprising the steps of: determining a device type for each of the first and second remote client devices;	See Claim 2 - element [A]. <i>In the embodiment having a plurality of local subscribers, “each local server has been replaced by a video server 176 which serves a plurality of local subscribers.” 9:63-64. The system operates otherwise as it would with a single subscriber.</i>
[B] using the device type of the first and second remote client devices to determine a format protocol for the requested service instance; and	See Claim 2 - element [B].
[C] reformatting the requested service instance prior to transmitting the	...and inherently, the service instance would be reformatted prior to transmission to the CD ROM writer or TV. The TV and the CD ROM writer use different

requested service instance.	signal formats.
Claim 14	Sections of Payton
[A] The method of claim 12, further comprising the steps of: determining an encryption scheme for the requested service instance based on the first and second remote client devices; and	
[B] prior to transmitting the requested service instance, encrypting the requested service instance.	
Claim 15	Sections of Payton
[A] The method of claim 14, further comprising the steps of: receiving a message along with the request from each of the first and second remote client devices, the message having device type information regarding the respective remote client device; and	
[B] determining from the device type information the encryption scheme for encryption of the requested service instance.	
Claim 16	Sections of Payton
[A] The method of claim 12, further comprising the steps of: receiving a message along with the request from each	

of the first and second remote client devices, the message having device type information regarding the respective remote client device;	
[B] transmitting a second message to a headend, the second message having the device type information included therein, wherein the headend determines the encryption scheme using at least the device type information.	
Claim 17	Sections of Payton
The method of claim 16, wherein the device type information from each of the first and second remote devices may require a same encryption scheme or a different encryption scheme.	
Claim 18	Sections of Payton
[A] The method of claim 12, further comprising the steps of: prior to transmitting the requested service instance to the first remote client device, determining whether the requested service instance is accessible within the communications device; and	3:25-27 (See Claim 6 - element [D]). 7:13-18 (See Claim 6 - element [E]).
[B] if the requested service	See Claim 6 - element [F].

instance is not accessible, sending a request to a headend for the requested service instance, wherein upon receipt of the service instance, transmitting the service instance from the communications device to the first remote client device.	
Claim 19	Sections of Payton
The method of claim 18, wherein the requested service instance is accessible if the service instance is stored in memory of the communications device.	<p><i>See Figure in anticipation of Claim 18 above:</i> If the instance is stored locally, it is accessible and is routed to the subscriber.</p> <p>“As shown in FIG. 3b, when a subscriber makes an on-demand request (step 74), the local server 28 polls the local storage 56 (step 75) to determine whether the requested item is stored locally (step 76). If it is stored locally, the item is sent to the decryption and decompression processor (step 77) 60 and then routed to the subscriber.” 7:13-19.</p>
Claim 20	Sections of Payton
The method of claim 18, further comprising the step of, prior to transmitting the service instance to the first remote client device, storing the received service instance in memory of the communications device.	<p><i>In the embodiment having a plurality of local subscribers, “each local server has been replaced by a video server 176 which serves a plurality of local subscribers.” 9:63-64. The system operates otherwise as it would with a single subscriber.</i></p> <p>6:11-17(See Claim 1 - element [E]). 8:11-25 (See Claim 7).</p>

With respect to Claim 1, Payton discloses **a method of transmitting service instances** (“in response to a subscriber’s request, the delivery system 22 delivers

the requested item” 4:51-53) **to a remote client device** (Payton’s “subscriber’s playback device 32” 4:53-54 is a remote client device) **in a local network** (Fig. 2 depicts that the local server 28 is networked with playback devices 32 and 65, also Fig. 8 depicts video server 176 connected over a local cable or wireless line 178 to multiple playback devices 32) , **including a communications device** (local server 28) **and one or more of said remote client devices** (playback devices 32, 65, see Fig. 2 and Fig. 8), **at a subscriber premises** (Payton’s local server is at a subscriber premise: “each subscriber is provided with a local storage device for storing ...items” Abstract), **comprising the steps of:**

receiving a service instance at the communications device of the subscriber premises from a headend (Payton’s local server receives service instances (items): “the digital transport system 26 broadcasts the digital items 36 from the central distribution server 24 to each of the local servers 28” 5:55-57), **wherein the service instance is stored in memory of the communications device** (“Each subscriber has a local server that downloads recommended and specifically requested items and stores them in a local storage device” 2:67-3:2);

receiving a request at the communications device (A subscriber’s request is received at Payton’s local server 28 because the local server 28 polls a local storage *in response to* a user’s request from a list, 3:22-28 (emphasis added), 7:13-15) **for the stored service instance** (some items on Payton’s recommended list are stored at the local storage, see 2:67-3:2 and 4:34, 36) **from the remote client**

device (Payton teaches that subscriber can make item selections from the remote client device using a control device 64, 6:26:31);

determining whether the remote client device is entitled to receive the requested stored service instance (Payton teaches the use of decryption prior to sending an item to the remote client device and therefore a determination is made about whether the remote client device is entitled to receive the requested stored service instance, 7:16-17, 10:1-3, 6, 13-17);

responsive to determining the remote client device is entitled to receive the requested stored service instance (Payton teaches that a requested stored service instance is decrypted and decompressed prior to sending to the remote client device. When decryption of a stored service instance fails, the output is decompressible, and therefore the service instance cannot be transmitted to the remote client device unless the remote client device is entitled to the service instance, i.e., the service instance is decryptable, see, 7:16-17, 10:1-3, 6, 13-17), **transmitting the requested service instance from the communications device to the remote client device** (“in response to a subscriber’s request, the delivery system 22 delivers the requested item” 4:51-53).

With respect to Claim 2, Payton further discloses:

determining a device type for the remote client device (Payton discloses that the format in which an item is sent depends on the device receiving the item. For example, “standard video and audio signals” are sent to a T.V. and “a digital

audio signal” is sent to a CD ROM writer. 6:18-25. The local server thus determines a device type of the remote client device before converting to the appropriate format and routing for sending to the device. 6:15-18, 8:19-20)

using the device type of the remote client device to determine a format protocol for the requested service instance (Payton discloses that the type of remote client device is used to determine the format protocol. For example, “standard video and audio signals” are sent to a T.V. and “a digital audio signal” is sent to a CD ROM writer. 6:18-25.);

reformatting the requested service instance prior to transmitting the requested service instance (Payton discloses sending “standard video and audio signals” to a T.V. and “a digital audio signal” to a CD ROM writer. 6:18-25.

Furthermore, Payton discloses that a decryption and decompression processor 60 decrypts, decompresses and converts the item from the digital format (in which the item is stored) to another format before transmitting to the remote client device. 6:13-14.).

Claims 3, 4 and 5 need 103 arguments

With respect to Claim 6, Payton discloses **a method of transmitting service instances** (“in response to a subscriber’s request, the delivery system 22 delivers the requested item” 4:51-53) **to a remote client device** (Payton’s “subscriber’s

playback device 32” 4:53-54 is a remote client device) **in a local network** (Fig. 2 depicts that the local server 28 is networked with playback devices 32 and 65, also Fig. 8 depicts video server 176 connected over a local cable or wireless line 178 to multiple playback devices 32) , **including a communications device** (local server 28) **and one or more of said remote client devices** (playback devices 32, 65, see Fig. 2 and Fig. 8), **at a subscriber premises** (Payton’s local server is at a subscriber premise: “each subscriber is provided with a local storage device for storing ...items” Abstract), **comprising the steps of:**

receiving a request at the communications device of the subscriber premises (A subscriber’s request is received at Payton’s local server 28 because the local server 28 polls a local storage *in response to* a user’s request from a list, 3:22-28 (emphasis added), 7:13-15) **for the stored service instance** (some items on Payton’s recommended list are stored at the local storage, see 2:67-3:2 and 4:34, 36) **from the remote client device** (Payton teaches that subscriber can make item selections from the remote client device using a control device 64, 6:26:31);

determining whether the remote client device is entitled to receive the requested stored service instance (Payton teaches the use of decryption prior to sending an item to the remote client device and therefore a determination is made about whether the remote client device is entitled to receive the requested stored service instance, 7:16-17, 10:1-3, 6, 13-17);

responsive to determining the remote client device is entitled to receive the requested stored service instance, determining whether the requested service instance is accessible within the communications device; (Payton determines accessibility of a requested service instance within the communication device because: “In response to a subscriber request, the local server first polls the local storage to see if the requested item is stored locally” 3:25-27; 5:36-38, 7:13-16), **if the requested service instance is accessible, transmitting the requested service instance from the communications device to the remote client device** (“in response to a subscriber’s request, the delivery system 22 delivers the requested item” 4:51-53).

if the requested service instance is not accessible, sending a request from the communications device to a headend for the requested service instance, (Payton’s local server sends a request to a central distribution server for the requested service instance using a back channel: “If the item is not immediately available the local server sends the request over the back channel to the central distribution server”, 3:28-30, 6:51-53)

wherein upon receipt of the service instance at the communications device, transmitting the service instance from the communications device to the remote client device (Payton’s local server transmits the service instance upon receipt: “As soon as the item is received locally, it is sent to a local playback device.” 3:31-33, 5:39-40).

With respect to Claim 12, Payton discloses **a method of transmitting service instances** (“in response to a subscriber’s request, the delivery system 22 delivers the requested item” 4:51-53) **to a plurality of remote client devices** (Payton’s “subscriber’s playback device 32” 4:53-54 is a remote client device. Payton also teaches a plurality of remote devices - e.g., a T.V., a computer, and a device that includes a CD-ROM writer, 6:20-24) **in a local network** (Fig. 2 depicts that the local server 28 is networked with playback devices 32 and 65, also Fig. 8 depicts video server 176 connected over a local cable or wireless line 178 to multiple playback devices 32), **including a communications device** (local server 28) **at a subscriber premises** (Payton’s local server is at a subscriber premise: “each subscriber is provided with a local storage device for storing ...items” Abstract), **comprising the steps of:**

receiving a first request for a service instance at the communications device of the subscriber premises (A subscriber’s request is received at Payton’s local server 28 because the local server 28 polls a local storage *in response to a* user’s request from a list, 3:22-28 (emphasis added), 7:13-15) **from a first remote client device** (Payton teaches that subscriber can make item selections from the remote client device using a control device 64, 6:26:31);

determining whether the first remote client device is entitled to receive the requested service instance (Payton teaches the use of decryption prior to sending an item to the remote client device and therefore a determination is made

about whether the remote client device is entitled to receive the requested stored service instance, 7:16-17, 10:1-3, 6, 13-17);

responsive to determining the first remote client device is entitled to receive the requested service instance (Payton determines accessibility of a requested service instance within the communication device because: “In response to a subscriber request, the local server first polls the local storage to see if the requested item is stored locally” 3:25-27; 5:36-38, 7:13-16), **transmitting the requested service instance from the communications device to the first remote client device** (“in response to a subscriber’s request, the delivery system 22 delivers the requested item” 4:51-53”;

receiving a second request for the service instance at the communications device from a second remote client device in the local network (A subscriber’s request is received at Payton’s local server 28 because the local server 28 polls a local storage *in response to* a user’s request from a list, 3:22-28 (emphasis added), 7:13-15. Further, Payton discloses that two remote devices can be operative in the network, e.g., a T.V., a computer and a CD-ROM writer 6:20-25.) ;

determining whether the second remote client device is entitled to receive the requested service instance (Payton teaches the use of decryption prior to sending an item to the remote client device and therefore a determination is

made about whether the remote client device is entitled to receive the requested stored service instance, 7:16-17, 10:1-3, 6, 13-17); and

responsive to determining the second remote client device is entitled to receive the requested service instance (Payton teaches that a requested stored service instance is decrypted and decompressed prior to sending to the remote client device. When decryption of a stored service instance fails, the output is decompressible, and therefore the service instance cannot be transmitted to the remote client device unless the remote client device is entitled to the service instance, i.e., the service instance is decryptable, see, 7:16-17, 10:1-3, 6, 13-17)), **transmitting the requested service instance from the communications device to the second remote client device** (in response to a subscriber's request, the delivery system 22 delivers the requested item" 4:51-53) **based upon local availability of the requested service instance within the communications device** (Payton's local server sends a request to a central distribution server for the requested service instance using a back channel: "If the item is not immediately available the local server sends the request over the back channel to the central distribution server", 3:28-30, 6:51-53).

***** **Payton Overview** *****

Payton discloses techniques for providing audio/video content ("digital items" or "items") to subscribers by predictively caching content from a

recommended list of items at a local server. Abstract. The content downloading is performed from a central distribution server over a cable, a fiber or a satellite network, preferably during off-peak hours, to reduce peak network bandwidth requirements. Abstract, Fig. 2. At a subscriber premise, the local server is connected with multiple playback devices having different capabilities (e.g., a computer or a television or a CD-ROM writer) in a local network. 1:50-63. Items that a user would like are called recommended items and are downloaded to local storage during off-peak hours and stored at the local storage. 4:34-38. Items specifically requested by the subscriber are also stored in the local storage device. 3:1-2. When the local server receives a request from a playback device, e.g., by the user interacting using a control device 64 such as a remote or a keyboard, the local server performs operations such as decryption, decompression, determination of whether the requested item is locally stored, and, if not, sending a message to the central distribution server in the cable or fiber or satellite network, for the item requested by the subscriber.

Payton transfers the requested item to the playback device by formatting based on the device type. For example, when the device is a television, the stored item is converted to standard video and audio signals and when the device is a CD-ROM device, the stored item is converted into a digital audio signal that can be recorded on a blank CD.

APPENDIX 27

CONFIDENTIAL - ATTORNEY WORK PRODUCT
Claim Chart for U.S. Patent No. 7,505,592 in view of HICKS (7,698,723)

Claim 1	Sections of Hicks
<p>[A] A method of transmitting service instances to a remote client device in a local network, including a communications device and one or more of said remote client devices, at a subscriber premises, comprising the steps of:</p>	<p><i>Hicks discloses a method of transmitting a service instances from a communication device to a remote client:</i> “The exemplary embodiments describe systems and methods for multimedia-on-demand services.” Abstract.</p> <p>“Upon receiving multimedia content, the BMG can transmit the multimedia content through the Ethernet switch over the twisted pair data network to an information appliance (e.g., a thin-client digital set-top box, an audio system, a wireless MP3 player, or a wireless electronic device), store the multimedia content for future access, or transmit and store coincidentally (e.g., simultaneously).” (information appliance = remote client device) 4:7-14.</p> <p>“For example, a television program broadcast by a CATV system can be received by BMG 100, and BMG can send that television program to a plurality of televisions 40 and/or computer 50 such that a user at an information appliance views the television program in real-time. BMG 100 can also store that television program on mass storage device 103...” 8:7-13.</p> <p>“As used to describe embodiments of the present invention, the term “multimedia” encompasses video, audio, audio-video, text, graphics, facsimile, data, animation, and combinations thereof. The BMG can deliver multimedia content to a wide range of information appliances, such as digital televisions, computers, sound systems, electronic book displays, and graphical data tablets.” 3:59-65.</p> <p>“Consumer access to many enhanced services--such as video/audio on demand, interactive TV, Web surfing, e-mail, electronic shopping and recording/storing/playback of broadcast programs-from each thin-client digital STB/TV in the home;” 5:55-59.</p> <p><i>Hicks discloses a local network including a communication device and one or more remote client devices:</i> “A core element of the system is a broadband multimedia gateway (BMG) that can operate both as a multimedia</p>

gateway and content server within a client/ server architecture. It contains an Ethernet switch that, in a typical embodiment, is capable of data communications of at least 100 Mbps per switch port. The BMG can receive video, audio and other forms of multimedia content from a variety of broadcasts (e.g., direct digital broadcast satellite TV, digital cable TV, terrestrial broadcast analog and/or digital TV),” (BMC = communication device) 3:50-58.

4:7-14 (See above).

“Because the enhanced functionality resides in the central BMG as opposed to peripheral thick-client digital STBs, a broad range of functionality, including record/store/playback of broadcast programs, video/audio on demand, interactive TV, Web surfing, e-mail and electronic shopping, is accessible from every thin-client digital STB in the home.” (information appliance = remote client device) 4:49-54.

“Within the digital residential entertainment system, the primary broadband data network can be supplemented and extended by the addition of plug-in modules for other lower bandwidth data networking technologies, such as Home Phoneline Networking Alliance (HomePNA) Version 2.0, HomeRF Shared Wireless Access Protocol (Home RF SWAP), IEEE 802.11, Bluetooth, and other similar technologies. For example, HomePNA Version 2.0 allows for the multiplexing of 10 Mbps of data over existing phone wiring in the home without interfering with analog telephony services operating over the same telephone wiring. HomeRF, IEEE 802.11 and Bluetooth are wireless data, or voice/data, technologies. Within the digital residential entertainment system, HomePNA, HomeRF, IEEE 802.11 and Bluetooth can principally be used for transmitting lower bandwidth multimedia content, such as audio content, as opposed to entertainment quality audio-video transmitted over the primary broadband data network. As newer technology emerges that improves the performance characteristics of HomePNA and “wireless” technology, entertainment quality audio-video can be supported over what is defined today as lower bandwidth technologies.” 3:26-47.

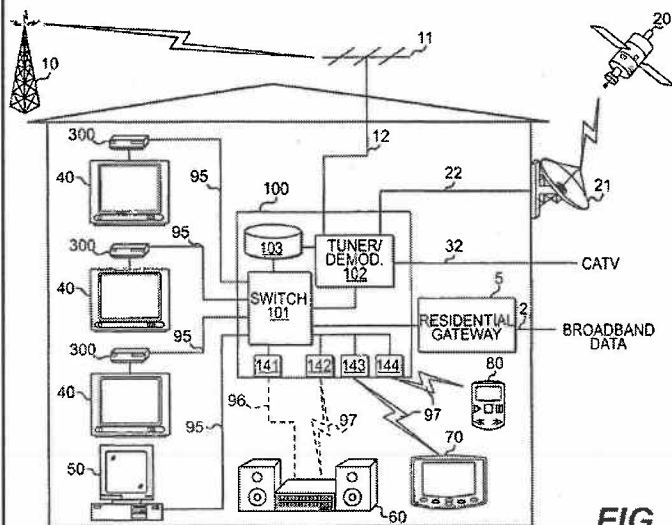


FIG. 1

“Examples of information appliances that can receive a digital information signal include a digital set top box 300, a television 40 (e.g., a television coupled to a digital set top box, a television including digital set top box functionality), a computer 50, an audio system 60, an electronic book device 70, an MP3 (MPEG Layer-3) player 80, an so on. Information appliances can be coupled to the data switch 101 via a high bandwidth (i.e., broadband) communication link 95,” 7:47-55.

Hicks discloses the network being located at a subscribers premises:

“According to the preferred embodiment of the present invention, a comprehensive digital residential entertainment system can provide access to multimedia content over an in-home broadband data network to a variety of information appliances.” 3:10-14.

“Systems and methods in accordance with the embodiments of the present invention disclosed herein can advantageously provide a digital residential entertainment system. In an embodiment, a digital residential entertainment system can provide access to multimedia content over an in-house broadband data network that is coupled to a data switch, a mass storage device and a variety of information appliances.” 17:24-30.

3:26-47 (See above).

	<p>“In the preferred embodiment of the present invention, the digital residential entertainment system is based on a client/server architecture.” 3:48-50.</p>
<p>[B] receiving a service instance at the communications device of the subscriber premises from a headend, wherein the service instance is stored in memory of the communications device;</p>	<p><i>Hicks discloses receiving a service instance at a communication device of the subscriber premises from a headend:</i></p> <p>“The BMG can receive video, audio and other forms of multimedia content from a variety of broadcasts (e.g., direct digital broadcast satellite TV, digital cable TV, terrestrial broadcast analog and/or digital TV), Intranet, and Internet sources. As used to describe embodiments of the present invention, the term “multimedia” encompasses video, audio, audio-video, text, graphics, facsimile, data, animation, and combinations thereof. The BMG can deliver multimedia content to a wide range of information appliances, such as digital televisions, computers, sound systems, electronic book displays, and graphical data tablets.” 3:55-65.</p> <p>“Multimedia content that is stored on the BMG can be accessed from any of the information appliances on the broadband home network.” 4:34-37.</p> <p>“The BMG can manage multiple demodulators/tuners to allow recording one or more broadcast programs while watching another broadcast program.” 4:28-30.</p> <p>“Tuner/demodulator 102 can be coupled to one or more of a plurality of multimedia transmission systems, where each multimedia transmission system transmits a plurality of transmission signals (e.g., audio, video, television, data, etc.). Examples of multimedia transmission systems include CATV, direct broadcast satellite TV, direct broadcast satellite radio, terrestrial broadcast TV, terrestrial broadcast radio, and so forth. Tuner/demodulator 102 can be coupled to a CATV system (e.g., a headend of a CATV system) via communications link 32 (e.g., a coaxial cable). A plurality of transmission signals from a direct broadcast satellite TV system including satellite 20 and satellite dish 21 can be received by tuner/ demodulator 102 via communications link 22. Also, tuner/demodulator 102 can be coupled to a terrestrial broadcast TV system via transmitter 10, antenna 11, and communications link 12.” 6:18-34.</p>

“In the direct digital broadcast satellite TV business and in the digital cable TV business, service providers have started to introduce enhanced digital STBs that are Web-enabled and include computer hard drives for supporting the recording, storage, and playback of broadcast content.” 1:46-50.

4:7-14 (See Claim 1 - element [A]).

“For example, a television program broadcast by a CATV system can be received by BMG 100, and BMG can send that television program to a plurality of televisions 40 and/or computer 50 such that a user at an information appliance views the television program in real-time. BMG 100 can also store that television program on mass storage device 103 so that a user can view the television program at a later time on one or more of the information appliances (e.g., televisions 40 and/or computer 50).” 8:7-16.

“Consumers’ homes typically include separate physical networks to support the distribution of video, audio, telephony, and data. For example, coaxial cable (such as quad-shielded RG6 coax), is often installed in homes for the distribution of audio/video (e.g., cable television (“CATV”), satellite broadcast television, local broadcast television) signals, while speaker wire is installed for the distribution of audio signals.” 1:12-18.

Hicks also discloses storing a service instance in memory of the communication device:

“Mass storage device 103 can be a hard disk drive, a magnetic storage device, an optical storage device, a magneto-optical storage device, or a combination thereof. The mass storage device 103 can store a digital information signal for subsequent playback and allows the BMG 100 to provide playback control (e.g., play, pause, rewind, fast forward, frame advance, etc.) of multimedia content (e.g., broadcast programs, movies, music, etc.).” 7:8-15.

“BMG 100 can also store that television program on mass storage device 103 so that a user can view the television program at a later time on one or more of the information appliances (e.g., televisions 40 and/or computer 50).” 8:12-16.

“...each tuner/demodulator can be coupled to a respective switch port of the Ethernet switch. In another embodiment, the multiple tuner/demodulators have a shared communication link to a switch port of the Ethernet switch. Upon receiving multimedia content, the BMG can transmit the multimedia content through the Ethernet switch over the twisted pair data network to an information appliance (e.g., a thin-client digital set-top box, an audio system, a wireless MP3 player, or a wireless electronic device), store the multimedia content for future access, or transmit and store coincidentally (e.g., simultaneously). The BMG includes a mass storage device (e.g., a computer hard drive) that can store multimedia content from broadcast sources, an Intranet or the Internet.” 4:3-16.

Figure 1 (See above).

“In an embodiment, a system can include a mass storage device, a processor and a memory. The mass storage device can receive and store a multimedia content item, and the memory can store a multimedia-on-demand data table and multimedia-on-demand instructions.” 2:23-27.

“Service providers will be able to download multimedia content, such as movies, to the mass storage device of the BMG.” 5:1-3.

“multimedia content can be stored in an encrypted format on the mass storage device.” 4:17-18.

“In accordance with an embodiment of the present invention, a service provider can download multimedia content items (e.g., movies, television programs, songs, albums, and so forth) to a multimedia-on-demand device (“MODD”) including a mass storage device that can store received multimedia content.” 12:66 - 13:4.

“Multimedia content downloading can be accomplished using a broadband data service, such as ADSL, a satellite direct multicast/broadcast service, a cable television service, a digital cable television service, a terrestrially broadcast television service, a wireless broadband data service, a wired broadband data service, and so on. Downloaded multimedia content items are stored on the mass storage device of the

	MODD.” 13:6-13.
[C] receiving a request at the communications device for the stored service instance from the remote client device;	<p><i>Hicks teaches receiving a request at a communication device for the stored service instance from the remote client device:</i></p> <p>“...a consumer can use an infrared remote control to select the content that he or she wants to view... the thin-client digital STB communicates with the BMG requesting that the multimedia content be delivered to the digital STB” 4:55-61.</p> <p>“Subsequently, a user can request that the movie be played back to an information appliance at the playback rate (e.g., real-time).” 15:23-25.</p> <p>“In an embodiment, a MODD can be part of a BMG that includes a data switch and is coupled to a plurality of information appliances via a plurality of broadband data links. A user at an information appliance of the plurality of information appliances can use a wireless infrared or RF remote control to access a BMG-generated Web page to determine what multimedia content items are currently stored on the MODD/BMG. After the consumer sends a usage instruction (e.g., playback instruction), the MODD/BMG can direct usage of the multimedia content item (e.g., begin streaming the selected multimedia content item to the information appliance).” 14:44-55.</p> <p>“In a movies-on-demand service, when a consumer wants to watch a movie, they would use their infrared remote control to access a Web page on the BMG Web server to determine what movies are currently stored on the BMG. After the consumer selects a movie for viewing, the BMG would begin streaming the selected movie out to the thin-client digital STB/TV for viewing.” 5:6-12.</p> <p>4:34-37 (See Claim 1 - element [B]).</p>
[D] determining whether the remote client device is entitled to receive the requested stored service instance; and	<p><i>Hicks discloses determining whether the remote client device is entitled to receive the requested stored service instance:</i></p> <p>“The signal processing circuit 120 can also be coupled to a decryption circuit/logic 127 that can decrypt and/or unscramble an encrypted and/or scrambled information signal, and a decoder 126 that can convert a digital</p>

information signal from one digital format to a second digital format. In a further embodiment, the decryption circuit/logic 127 is coupled to a smartcard reader 129 to support CAS functionality.” 9:8-15.

“In an embodiment, information signals can be encrypted and/or decrypted by cipher/decipher logic 168. Cipher/decipher logic 168 can decrypt and/or encrypt information signals according to encryption/copy protection protocols such as an Analog CPS (Copy Protection System) (e.g., a Macrovision protocol), CGMS (Copy Guard Management System), CSS (Content Scrambling System), CPPM (Content Protection for Prerecorded Media), CPRM (Content Protection for Recordable Media), DCPS (Digital Copy Protection System), DTCP (Digital Transmission Content Protection), and so forth. An information signal received from the auxiliary multimedia input 166 can be stored-either encrypted or unencrypted on the mass storage device 103 and/or sent to one or more information appliances coupled to data switch/router 105.” 9:22-36.

“Only authorized and ultimately authenticated smart cards inserted into the entertainment system (e.g., into card reader 129, coupled to input/output 137) can open locally accessible debug operations. “Consumer” smart cards are authorized for services designed and developed for entertainment content delivery. Thus, an embodiment of the present invention has real-time pay-per-view and multimedia delivery support. In a further embodiment, pay-per-view events can be authorized at least in part with a CAS smartcard device.” 10:58-67.

“In another embodiment, a multimedia content item usage indicator field can store one or more multimedia content item usage indicators corresponding to the multimedia content item (e.g., an indicator that the item has not been played or purchased, that the item was purchased, that the item was licensed, and so forth).” 14:33-38.

“In an embodiment, multimedia content can be stored in an encrypted format on the mass storage device. Thin-client information appliances, such as digital STBs, can include decoding and/or deciphering capabilities. Encryption of multimedia content can ensure that proprietary and/or

copyrighted material is protected as it is transmitted across the residential broadband data network. Conditional access systems (“CAS”) using smartcard technology, such as those manufactured by NagraCard S.A. of Cheseaux, Switzerland and NDS Group PLC of the United Kingdom, can be integrated in the entertainment system.” 4:17-27.

“TC DSTB 300 can receive a digital information signal from the data switch and process the digital information signal for output as an audio and/or video signal. For example, TC DSTB 300 can include decryption logic 320 coupled to the Ethernet interface 310 via a bus 315, and the decryption logic 320 can decrypt digital information signals that are sent by the data switch in an encrypted and/or protected format.” 11:25-29.

[E] responsive to determining the remote client device is entitled to receive the requested stored service instance, transmitting the requested service instance from the communications device to the remote client device.

3:55-65 (See Claim 1 - element [B]).
4:34-37 (See Claim 1 - element [B]).
9:22-36 (See Claim 1 - element [D]).

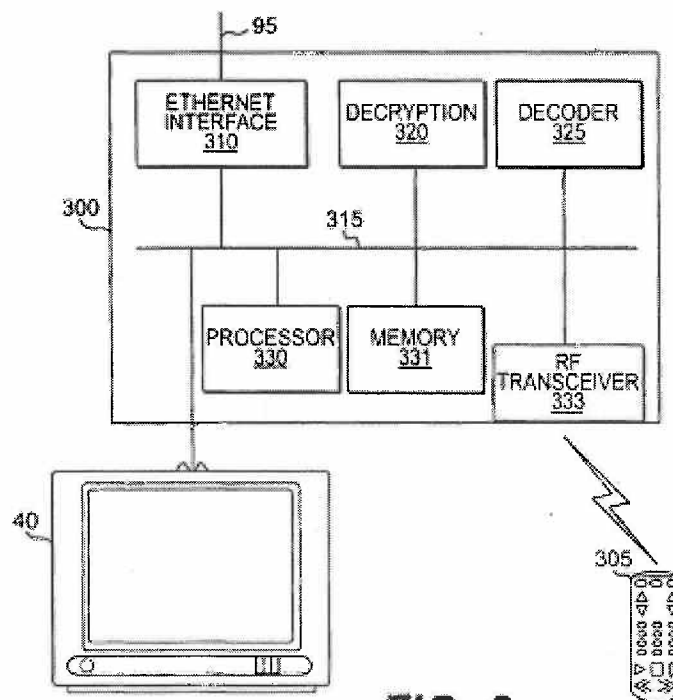


FIG. 3

“FIG. 3 is an illustration of a thin-client digital set top box according to a preferred embodiment of the present invention.” 11:1-2.

“For example, TC DSTB 300 can include decryption logic 320 coupled to the Ethernet interface 310 via a bus 315, and

	<p>the decryption logic 320 can decrypt digital information signals that are sent by the data switch in an encrypted and/or protected format. Decoder logic 325 can also be included in TC DSTB 300 to convert a digital information signal from a first digital format (e.g., a transmission format, a compressed format) to a second digital format (e.g., a display format).” 11:25-33.</p> <p><i>Hicks further discloses transmitting the requested service instance from the communication device to the remote client device in response to determining remote device is entitled:</i></p> <p>4:7-14 (See Claim 1 - element [A]). 14:46-55 (See Claim 1 - element [C]).</p> <p>“In an embodiment in which a MODSP offers a pay-per-view movie service, a user can opt to receive the pay-per-view service (e.g., subscribe to the service) or may receive the pay-per-view service because it is a system default service that each user receives as part of using the system (e.g., it is a bundled component of a digital cable service, a direct broadcast satellite television service, and so on).” 13:29-36.</p> <p>“After the consumer makes a selection, the thin-client digital STB communicates with the BMG requesting that the digital multimedia content be delivered to the digital STB. When the consumer selects playing of a broadcast satellite television channel, for example, the BMG can tune a demodulator/tuner to the selected broadcast channel and begin streaming the selected MPEG video stream through the Ethernet switch and over the twisted pair wiring to the digital STB where the video steam is decoded and displayed on the TV.” 4:58-67.</p>
Claim 2	Sections of Hicks
<p>[A] The method of claim 1, further comprising the steps of: determining a device type for the remote client device;</p>	<p>“Examples of information appliances that can receive a digital information signal include a digital set top box 300, a television 40 (e.g., a television coupled to a digital set top box, a television including digital set top box functionality), a computer 50, an audio system 60, an electronic book device 70, an MP3 (MPEG Layer-3) player 80, an so on.” 7:47-53.</p> <p>“...a decoder 126 that can convert a digital information</p>

	signal from one digital format to a second digital format.” 9:11-14.
[B] using the device type of the remote client device to determine a format protocol for the requested service instance; and	(See all of Claim 2 - element [A]).
[C] reformatting the requested service instance prior to transmitting the requested service instance.	(See all of Claim 2 - element [A]).
Claim 3	Sections of Hicks
[A] The method of claim 1, further comprising the steps of: determining an encryption scheme for the requested service instance based on the remote client device; and	“Encryption of multimedia content can ensure that proprietary and/or copyrighted material is protected as it is transmitted across the residential broadband data network. Conditional access systems (“CAS”) using smartcard technology, such as those manufactured by NagraCard S.A. of Cheseaux, Switzerland and NDS Group PLC of the United Kingdom, can be integrated in the entertainment system.” 4:21-27.
[B] prior to transmitting the requested service instance, encrypting the requested service instance.	9:23-36 (See Claim 1 - element [E]).
Claim 4	Sections of Hicks
[A] The method of claim 3, further comprising the steps of: receiving a message along with the request from the remote client device, the message having device type information regarding the remote client device; and	“Cipher/decipher logic 168 can decrypt and/or encrypt information signals according to encryption/copy protection protocols such as an Analog CPS (Copy Protection System) (e.g., a Macrovision protocol), CGMS (Copy Guard Management System), CSS (Content Scrambling System), CPPM (Content Protection for Prerecorded Media), CPRM (Content Protection for Recordable 30 Media), DCPS (Digital Copy Protection System), DTCP (Digital Transmission Content Protection), and so forth.” 9:24-32.
[B] determining from the device type information the encryption scheme for encryption of the requested	(See Claim 4 - element [A]).

service instance.	
Claim 5	Sections of Hicks
[A] The method of claim 3, further comprising the steps of: receiving a message along with the request from the remote client device, the message having device type information regarding the remote client device;	(See Claim 4 - element [A]).
[B] transmitting a second message to the headend, the second message having the device type information included therein, wherein the headend determines the encryption scheme using at least the device type information.	
Claim 6	Sections of Hicks
[A] A method of transmitting service instances to a remote client device in a local network, including a communications device and one or more of said remote client devices, at a subscriber premises, comprising the steps of:	(See all of Claim 1 - element [A]).
[B] receiving a request at the communications device of the subscriber premises for a service instance from the remote client device;	(See all of Claim 1 - element [C]).
[C] determining whether the remote client device is entitled to receive the requested service instance;	(See all of Claim 1 - element [D]).

<p>[D] responsive to determining the remote client device is entitled to receive the requested service instance, determining whether the requested service instance is accessible within the communications device;¹</p>	<p>(See all of Claim 1 - element [E]).</p>
<p>[E] if the requested service instance is accessible, transmitting the service instance from the communications device to the remote client device; and</p>	<p>4:34-37 (See Claim 1 - element [B]). 8:12-16 (See Claim 1 - element [B]).</p> <p>“To access the automatically downloaded content, the user can instruct the MODD to display a listing of the “Top Ten Movies” that are stored on the mass storage device of the MODD and select one for playback.” 13:49-52.</p> <p>“For example, FIG. 5 shows a data table in accordance with an embodiment of the present invention. A MODD can include a data table 500 (e.g., stored on a mass storage device, stored in non-volatile memory, etc.) Data table 500 can include a plurality of data records 501. Each data record 501 can correspond to a multimedia content item stored on the mass storage device.” 14:4-10.</p> <p>“For example, multimedia content item usage indicator field 520 can store an indication regarding whether the corresponding multimedia content item has been played (e.g., played for viewing, played for listening), and multimedia content item usage indicator field 520 can store an indication as to whether the corresponding multimedia content item has been purchased (e.g., copied to a portable non-volatile storage medium such as a recordable CD-ROM or DVD, sent to an information appliance that can store the multimedia content item, changed from a temporary file to a permanent file resident on the mass storage device, copied to a separate mass storage device for archival purposes, etc.).” 14:21-33.</p> <p>“As consumer data services offering significantly higher bandwidth to the home become available, such as fiber optical networks extending into the home, it will be possible</p>

¹ The determination of whether the service instance is accessible within the communication device or not is inherently described by Hicks because different viewers can be watching programs that are either “live” from the CATV network or are transmitted from a recording on the mass storage device.

	<p>to install 35 a BMG, or a system with comparable functionality, outside of the home in a network-based platform.” 5:33-37.</p> <p>“After the consumer selects a movie for viewing, the BMG would begin streaming the selected movie out to the thin-client digital STB/TV for viewing.” 5:10-12.</p>
<p>[F] if the requested service instance is not accessible, sending a request from the communications device to a headend for the requested service instance, wherein upon receipt of the service instance at the communications device, transmitting the service instance from the communications device to the remote client device.</p>	<p><i>Hicks discloses that is the service instance is not accessible, sending a request from the communication device to a headend for the requested service:</i></p> <p>“The enhanced digital STBs can provide for pay-per-view movie delivery, but such services typically require the consumer to select a pay-per-view that is pre-scheduled for a particular time. For example, the consumer may have the choice of watching the pay-per-view movie at 8:00p.m. or at 9:00p.m. If the consumer, for example, wanted to watch the pay-per-view movie beginning at 7:00p.m., she nevertheless would have to wait until 8:00 p.m. to begin viewing the pay-per-view movie.” (PPV) 1:53-61.</p> <p>10:63-67 (See Claim 1 - element [D]).</p> <p>“In an embodiment, a MODD can be part of a BMG that includes a plurality of tuner/demodulators and a data switch. Multimedia content downloading can be accomplished using a broadband data service, such as ADSL, a satellite direct multicast/broadcast service, a cable television service, a digital cable television service, a terrestrially broadcast television service, a wireless broadband data service, a wired broadband data service, and so on. Downloaded multimedia content items are stored on the mass storage device of the MODD. Each stored multimedia content item can be identified by a multimedia content item identifier, and use of the multimedia content item (e.g., playback, purchase of a copy, licensing of a copy, etc.) can be indicated by a multimedia content item usage indicator. Usage of the multimedia content item can be reported to the multimedia-on-demand service provider (“MODSP”) by transmitting a usage message to the MODSP. For example, a usage message can be based at least in part on the multimedia content item usage indicator and report that a subscriber viewed a movie, listened to a song, copied an album to non-volatile medium (e.g., a recordable CD-ROM, a recordable</p>

DVD).” (MODD service can augment the classical services described earlier (which include traditional PPV) 13:4-24.

“In an embodiment in which a MODSP offers a pay-per-view movie service, a user can opt to receive the pay-per-view service (e.g., subscribe to the service) or may receive the pay-per-view service because it is a system default service that each user receives as part of using the system (e.g., it is a bundled component of a digital cable service, a direct broadcast satellite television service, and so on). But the user need not select or direct the downloading of an individual multimedia content item. The MODSP can, however, automatically send the plurality of multimedia content items based on a subscriber profile or a system profile, and the user can modify or update such profiles (e.g., to select a particular genre of movies, music, content, and so on). 13:29-42.

“In another embodiment of the present invention, the MODD can indicate that a multimedia content item is available for playback prior to storing the entirety of the multimedia content item.” (playback can begin before download has completed) 15:26-29.

“In an embodiment of the present invention, a user at any appliance through a Web-based graphical user interface can instruct the processor to record any broadcast program onto mass storage for subsequent viewing and/or listening from a multiplicity of appliances. The user can choose to have the broadcast program.” 17:17-23.

“The multimedia-on-demand instruction can be executed by the processor and can include instructions to automatically receive the multimedia content item and send a multimedia-on-demand usage message, where the multimedia-on-demand usage message is based at least in part on the multimedia content usage indicator.” 2:36-41.

“Because the enhanced functionality resides in the central BMG as opposed to peripheral thick-client digital STBs, a broad range of functionality, including record/store/playback of broadcast programs, video/audio on demand, interactive TV, Web surfing, e-mail and electronic shopping, is accessible from every thin-client digital STB in the home.

For example, to view broadcast video content, a consumer can use an infrared remote control to select the content that he or she wants to view by utilizing a broadcast program guide, a search function, entering a channel number, and so on. After the consumer makes a selection, the thin-client digital STB communicates with the BMG requesting that the digital multimedia content be delivered to the digital STB.” (discussion surrounding web opens up the knowledge OSITA has of caching web pages and refreshing if necessary.) 4:49-61.

“An example of overlay processing is presenting MPEG2 or other digital information in a multiple layer format. In another embodiment, a service application running on a BMG system can support transparent layers such as, for example, overlaying a Web page on top of a TV program image used for interactive TV services.” (mentions interactive TV applications and a web page overlaid on a TV signal. In this case, data could have been downloaded “in advance” to the STB or it could be accessed as needed over the internet) 7:1-6.

Hicks discloses wherein upon receipt of the service instance at the communications device, transmitting the service instance from the communications device to the remote client device:

“Upon receiving multimedia content, the BMG can transmit the multimedia content through the Ethernet switch over the twisted pair data network to an information appliance (e.g., a thin-client digital set-top box, an audio system, a wireless MP3 player, or a wireless electronic device), store the multimedia content for future access, or transmit and store coincidentally (e.g., simultaneously). The BMG includes a mass storage device (e.g., a computer hard drive) that can store multimedia content from broadcast sources, an Intranet or the Internet.” 4:7-16.

Figure 4 (outlines the steps towards deciding to transmit the service instance).

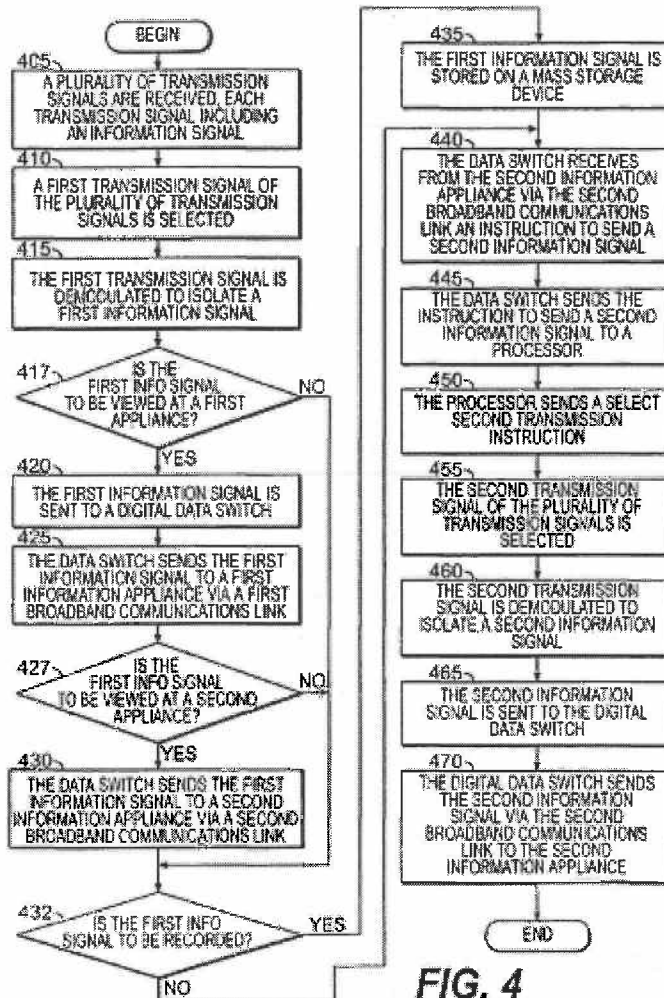


FIG. 4

4:61-67. (See Claim 1 - element [E]).

Claim 7	Sections of Hicks
The method of claim 6, further comprising the step of, prior to transmitting the service instance to the remote client device, storing the received service instance in memory of the communications device.	2:23-26 (See Claim 1 - element [B]). 4:17-18 (See Claim 1 - element [B]). 5:1-3 (See Claim 1 - element [B]). 12:66 - 13:4 (See Claim 1 - element [B]). 13:6-13 (See Claim 1 - element [B]).
Claim 8	Sections of Hicks
[A] The method of claim 6, further comprising the steps of:	(See all of Claim 2 - element [A]).

determining a device type for the remote client device;	
[B] using the device type of the remote client device to determine a format protocol for the requested service instance; and	(See all of Claim 2 - element [A]).
[C] reformatting the requested service instance prior to transmitting the requested service instance.	(See all of Claim 2 - element [A]).
Claim 9	Sections of Hicks
[A] The method of claim 6, further comprising the steps of: determining an encryption scheme for the requested service instance based on the remote client device; and	4:21-27 (See Claim 3 - element [A]).
[B] prior to transmitting the requested service instance, encrypting the requested service instance.	9:23-36 (See Claim 1 - element [D]).
Claim 10	Sections of Hicks
[A] The method of claim 9, further comprising the steps of: receiving a message along with the request from the remote client device, the message having device type information regarding the remote client device; and	(See Claim 4 - element [A]).
[B] determining from the device type information the encryption scheme for encryption of the requested service instance.	(See Claim 4 - element [A]).

Claim 11	Sections of Hicks
<p>[A] The method of claim 9, further comprising the steps of: receiving a message along with the request from the remote client device, the message having device type information regarding the remote client device;</p>	<p>(See Claim 4 - element [A]).</p>
<p>[B] transmitting a second message to the headend, the second message having the device type information included therein, wherein the headend determines the encryption scheme using at least the device type information.</p>	
Claim 12	Sections of Hicks
<p>[A] A method of transmitting service instances to a plurality of remote client devices in a local network, including a communications device, at a subscriber premises, comprising the steps of:</p>	<p>“The exemplary embodiments describe systems and methods for multimedia-on-demand services.” Abstract.</p> <p>“Embodiments of the present invention relate to multimedia-on-demand services that can be provided by automatic downloading of selections of multimedia content to a mass storage device.” 2:14-17.</p> <p>3:10-14 (See Claim 1 - element [A]).</p>
<p>[B] receiving a first request for a service instance at the communications device of the subscriber premises from a first remote client device;</p>	<p>“After the consumer sends a usage instruction (e.g., playback instruction), the MODD/BMG can direct usage of the multimedia content item.” 14:51-53.</p> <p>4:34-37 (See Claim 1 - element [B]). 4:55-61 (See Claim 1 - element [C]).</p>
<p>[C] determining whether the first remote client device is entitled to receive the requested service instance;</p>	<p>4:17-27 (See Claim 1 - element [D]). 11:25-29 (See Claim 1 - element [D]).</p>

<p>[D] responsive to determining the first remote client device is entitled to receive the requested service instance, transmitting the requested service instance from the communications device to the first remote client device;</p>	<p>4:58-67 (See Claim 1 - element [E]).</p> <p>“Audio content, such as an albums-on-demand service, could be implemented in a similar manner to allow playback and purchasing of audio content.” 5:18-20.</p> <p>“A user can opt to receive the pay-per-view service (e.g., subscribe to the service).” 13:30-32.</p>
<p>[E] receiving a second request for the service instance at the communications device from a second remote client device in the local network;</p>	<p>“For example, a viewer at a first television may view a first CATV program, and another viewer at a second television may view a second CATV program. Further, a viewer can view a first CATV program while a second CATV program is recorded to the mass storage device.” 8:48-52.</p> <p>3:10-14 (See Claim 1 - element [A]).</p> <p>14:51-53 (See Claim 12 - element [B]).</p> <p>4:34-37 (See Claim 1 - element [B]).</p> <p>4:55-61 (See Claim 1 - element [C]).</p>
<p>[F] determining whether the second remote client device is entitled to receive the requested service instance; and</p>	<p>4:17-27 (see Claim 1 - element [D]).</p> <p>11:25-29 (see Claim 1 - element [D]).</p>
<p>[G] responsive to determining the second remote client device is entitled to receive the requested service instance, transmitting the requested service instance from the communications device to the second remote client device based upon local availability of the requested service instance within the communications device.</p>	<p>4:58-67 (See Claim 1 - element [E]).</p> <p>5:18-20 (See Claim 12 - element [D]).</p> <p>13:30-32 (See Claim 12 - element [D]).</p>
<p>Claim 13</p>	<p>Sections of Hicks</p>
<p>[A] The method of claim 12, further comprising the steps of: determining a device type for</p>	<p>(See all of Claim 2 - element [A]).</p>

each of the first and second remote client devices;	
[B] using the device type of the first and second remote client devices to determine a format protocol for the requested service instance; and	(See all of Claim 2 - element [A]).
[C] reformatting the requested service instance prior to transmitting the requested service instance.	(See all of Claim 2 - element [A]).
Claim 14	Sections of Hicks
[A] The method of claim 12, further comprising the steps of: determining an encryption scheme for the requested service instance based on the first and second remote client devices; and	4:21-27 (See Claim 3 - element [A]).
[B] prior to transmitting the requested service instance, encrypting the requested service instance.	9:23-36 (See Claim 1 - element [D]).
Claim 15	Sections of Hicks
[A] The method of claim 14, further comprising the steps of: receiving a message along with the request from each of the first and second remote client devices, the message having device type information regarding the respective remote client device; and	(See Claim 4 - element [A]).
[B] determining from the	(See Claim 4 - element [A]).

device type information the encryption scheme for encryption of the requested service instance.	
Claim 16	Sections of Hicks
[A] The method of claim 12, further comprising the steps of: receiving a message along with the request from each of the first and second remote client devices, the message having device type information regarding the respective remote client device;	(See Claim 4 - element [A]).
[B] transmitting a second message to a headend, the second message having the device type information included therein, wherein the headend determines the encryption scheme using at least the device type information.	
Claim 17	Sections of Hicks
The method of claim 16, wherein the device type information from each of the first and second remote devices may require a same encryption scheme or a different encryption scheme.	4:17-27 (See Claim 1 - element [D]).
Claim 18	Sections of Hicks
[A] The method of claim 12, further comprising the steps of: prior to transmitting the requested service instance to	8:48-52 (See Claim 12 - element [E]).

the first remote client device, determining whether the requested service instance is accessible within the communications device; and	
[B] if the requested service instance is not accessible, sending a request to a headend for the requested service instance, wherein upon receipt of the service instance, transmitting the service instance from the communications device to the first remote client device.	4:61-67. (See Claim 1 - element [E]).
Claim 19	Sections of Hicks
The method of claim 18, wherein the requested service instance is accessible if the service instance is stored in memory of the communications device.	4:34-37 (See Claim 1 - element [B]). 5:10-12 (See Claim 6 - element [E]).
Claim 20	Sections of Hicks
The method of claim 18, further comprising the step of, prior to transmitting the service instance to the first remote client device, storing the received service instance in memory of the communications device.	2:23-26 (See Claim 1 - element [B]). 4:17-18 (See Claim 1 - element [B]). 5:1-3 (See Claim 1 - element [B]). 12:66 - 13:4 (See Claim 1 - element [B]). 13:6-13 (See Claim 1 - element [B]).