(19)日本国特許庁(JP)　(12)特　許　公　報(B2)　(11)特許番号
特許第4064060号
(P4064060)
(45)発行日　平成20年3月19日(2008.3.19)　(24)登録日　平成20年1月11日(2008.1.11)

(51)Int.Cl.　　F I
G06Q 30/00 (2006.01)　　G06F 17/60 326
G06Q 10/00 (2006.01)　　G06F 17/60 504

請求項の数 6　(全 51 頁)

(21)出願番号	特願2000-550046 (P2000-550046)	(73)特許権者	500525324
(86)(22)出願日	平成11年5月14日 (1999.5.14)		ユニキャスト・コミュニケーションズ・コーポレイション
(65)公表番号	特表2002-516437 (P2002-516437A)		アメリカ合衆国、１００１９　ニュー・ヨーク州、ニュー・ヨーク、フィフス・アベニュ、６５０、ナインス・フロアー
(43)公表日	平成14年6月4日 (2002.6.4)	(74)代理人	100064746
(86)国際出願番号	PCT/US1999/010707		弁理士　深見　久郎
(87)国際公開番号	WO1999/060504	(74)代理人	100085132
(87)国際公開日	平成11年11月25日 (1999.11.25)		弁理士　森田　俊雄
審査請求日	平成15年2月21日 (2003.2.21)	(74)代理人	100083703
(31)優先権主張番号	09/080,165		弁理士　仲村　義平
(32)優先日	平成10年5月15日 (1998.5.15)	(74)代理人	100091409
(33)優先権主張国	米国(US)		弁理士　伊藤　英彦
(31)優先権主張番号	09/237,718		
(32)優先日	平成11年1月26日 (1999.1.26)		
(33)優先権主張国	米国(US)		

最終頁に続く

(54)【発明の名称】参照ウェブページに埋め込まれた広告タグをもちいて、ブラウザにより開始される、ユーザには見えないネットワーク分散隙間ウェブ公告を実施するための技術

(57)【特許請求の範囲】

【請求項１】

プロセッサ（３４０）と、前記プロセッサに接続されコンピュータ実行可能命令（４００）および第１のウェブページ（３５）の両方を記憶するメモリ（３３０）とを有するコンピュータ（５）において用いるための方法であって、前記第１のウェブページはページコンテンツと、情報オブジェクトを選択することを求めるために埋込まれたコード（４０）とを表わす複数のコンピュータ読取可能命令を有し、前記方法は、前記実行可能命令に応答しかつ前記コードを実行した結果として、プロセッサによって行なわれる以下のステップ、すなわち、

ネットワークを介して管理サーバ（２０）に前記情報オブジェクトを選択することを求める要求を通信インターフェースを用いて通信するステップと、

前記コンピュータがそのコンピュータと関連して動作する出力装置（３８０）を介してユーザに対し前記第１のウェブページを表示したまま、前記管理サーバによって選択された情報オブジェクトを後で前記プロセッサにより表示すべく用いられることになる複数のファイルを、前記要求に応答したネットワークサーバ（１５）または前記管理サーバから連続的にダウンロードするステップとを含み、前記プロセッサによって行なわれるステップはさらに、

前記コンピュータにより検出される、前記第１のウェブページから次の連続するウェブページに移行するための、次の隙間期間の開始を意味する、ユーザにより開始されるイベント（５１０）に応答して、前記管理サーバによって選択された情報オブジェクトを前記

プロセッサにより表示すべく用いられることになるファイルのダウンロードをサスペンドし、ダウンロードされたファイルを処理して前記オブジェクトを前記出力装置を介してユーザに対し前記隙間期間中に表示するステップを含み、
前記コードは前記情報オブジェクトへの参照を含まない、方法。
【請求項２】
情報オブジェクトを選択することを求めるために埋込まれたコード（４０）を含む第１のウェブページ（３５）に応答して情報オブジェクトを表示するための装置（５）であって、
プロセッサ（３４０）と、
前記プロセッサに接続されコンピュータ実行可能命令（４００）および前記第１のウェブページの両方を記憶するメモリ（３３０）とを含み、前記第１のウェブページはページコンテンツおよび前記埋込まれたコードを表わす複数のコンピュータ読取可能命令を有し、
前記プロセッサは、前記実行可能命令に応答しかつ前記コードをブラウザを用いて実行することにより、
ネットワークを介して前記管理サーバ（２０）に前記情報オブジェクトを選択することを求める要求を通信インターフェースを用いて通信し、
前記コンピュータがそのコンピュータと関連して動作する出力装置（３８０）を用いてユーザに対し前記第１のウェブページを表示したまま、前記管理サーバによって選択された情報オブジェクトを後で前記プロセッサにより表示すべく用いられることになる複数のファイルを、前記要求に応答したネットワークサーバ（１５）または前記管理サーバから通信インターフェースを用いて連続的にダウンロードし、前記プロセッサはさらに、
前記コンピュータにより検出される、前記第１のウェブページから次の連続するウェブページに移行するための、次の隙間期間の開始を意味する、ユーザにより開始されるイベントに応答して、前記管理サーバによって選択された情報オブジェクトを前記プロセッサにより表示すべく用いられることになるファイルのダウンロードをサスペンドし、ダウンロードされたファイルを処理して前記情報オブジェクトを前記出力装置を用いてユーザに対し前記隙間期間中に表示し、
前記コードは前記情報オブジェクトへの参照を含まない、装置。
【請求項３】
第１のウェブページ（３５）に応答して情報オブジェクトを表示するための装置（５）であって、
前記第１のウェブページは、ページコンテンツと、前記情報オブジェクトを選択することを求めるために埋込まれたコードとの両方を含み、前記装置は、
前記ページコンテンツおよびコード（４０）を表わすコンピュータ実行可能命令（４００）を記憶するメモリ（３３０）と、
出力装置（３８０）と、
前記コンピュータ実行可能命令に応答し、かつ前記ページ上で前記命令を処理している間前記埋込まれたコードをブラウザを用いて実行することにより、第１のサーバ（１５）からエージェント（５６、４２０）を通信インターフェースを用いてダウンロードし、次いで前記エージェントを実行（５８）し、前記エージェントを用いて、
前記第１のウェブページが前記出力装置において表示されている間に、前記情報オブジェクトの一部、１つの情報ファイルにアクセス可能なネットワークアドレス、および前記情報オブジェクトを適切に表示するのに必要な関連の構成情報を含む少なくとも１つの予め規定された情報ファイルを特定する、前記情報オブジェクトに関連づけられる明示されたファイル（４３３）を、前記情報オブジェクトを選択する第２のサーバ（２０）から通信インターフェースを用いて前記メモリにダウンロードし、かつ前記明示されたファイルに特定されるとおりの少なくとも１つの付加的なファイルを対応のファイルサーバから前記メモリに通信インターフェースを用いてダウンロードし、
前記第１のウェブページから第２のウェブページへの移行を開始するための、隙間期

間の開始を意味する、ユーザにより開始されるイベント（５１０）を入力インターフェースを用いて検出し、

前記イベントの発生に応答して、

前記明示されたファイルに特定されるさらなるファイルのダウンロードを、そのさらなるファイルのどのダウンロードもその時生じている程度で停止し、

以前にダウンロードされており、現在表示される準備ができている情報オブジェクトに対するファイルを処理することにより、以前にダウンロードされた情報オブジェクトを前記隙間期間中に出力装置を用いてユーザに表示する、前記メモリが接続されたプロセッサ（３４０）とを含む、装置。

【請求項４】

プロセッサ（３４０）と、前記プロセッサに接続されページコンテンツおよびコード（４０）を表わすコンピュータ実行可能命令（４００）を記憶するメモリ（３３０）と、前記プロセッサに応答する出力装置（３８０）とを有するコンピュータ（５）において、第１のウェブページ（３５）に応答して情報オブジェクトを表示するための方法であって、

前記第１のウェブページは、ページコンテンツと、前記情報オブジェクトを選択することを求めるために埋込まれたコードとの両方を含み、前記方法は、

前記第１のウェブページ上で前記命令を処理する間前記埋込まれたコードを前記プロセッサがブラウザを用いて実行することにより、第１のサーバ（１５）からエージェント（４２０）を通信インターフェースを用いてダウンロードし次いで前記エージェントを実行するステップと、前記プロセッサが、前記エージェントを用いて、

前記第１のウェブページの前記出力装置上における表示中に、前記情報オブジェクトの一部、１つの情報ファイルにアクセス可能なネットワークアドレス、および前記情報オブジェクトを適切に表示するのに必要な関連の構成情報を含む、少なくとも１つの予め規定された情報ファイルを特定する、前記情報オブジェクトに関連づけられる明示されたファイル（４３３）を、前記情報オブジェクトを選択する第２のサーバ（２０）から前記メモリに通信インターフェースを用いてダウンロードし、かつ、前記明示されたファイルに特定されるとおりの少なくとも１つの付加的なファイルを対応のファイルサーバから前記メモリに通信インターフェースを用いてダウンロードするステップと、

前記第１のウェブページから第２のウェブページへの移行を開始するための、隙間期間の開始を意味する、ユーザにより開始されるイベント（５１０）を入力インターフェースを用いて検出するステップと、

前記イベントの発生に応答して、

前記明示されたファイルに特定されるさらなるファイルのダウンロードを、そのさらなるファイルのどのダウンロードもその時生じている程度で停止するステップと、

以前にダウンロードされており、現在表示される準備ができている情報オブジェクトに対するファイルを処理することにより、以前にダウンロードされた情報オブジェクトを前記隙間期間中に出力装置を用いてユーザに表示するステップとを含む、方法。

【請求項５】

埋込まれたコード（４０）を含む第１のウェブページ（３５）に応答して情報オブジェクトを表示する際に用いる装置（５）であって、

前記第１のウェブページは、ページコンテンツと、情報オブジェクトを選択することを求めるために埋込まれたコードとを表わす複数のコンピュータ読取可能命令を有し、前記装置は、

コンピュータ実行可能命令および前記第１のウェブページの両方を記憶するメモリ（３３０）と、

出力装置（３８０）と、

前記実行可能命令に応答しかつウェブブラウザ（７）を用いて前記コードを実行することにより、前記コンピュータが前記第１のウェブページをユーザに対して出力装置を介し表示している間に、第２のサーバ（２０）によって選択された情報オブジェクトを後で表示すべく用いることになる少なくとも１つのファイルをダウンロードし、ユーザにより発

生されるクリックストリームを入力インターフェースを用いて監視することにより、前記第１のウェブページから次の連続するウェブページへ移行するユーザ動作を意味する、次の隙間期間の開始を意味するユーザナビゲーションイベント（５１０）を検出し、前記ユーザナビゲーションイベントに応答して、前記１つのファイルを処理することにより、前記第２のサーバによって選択された情報オブジェクトを前記出力装置を用いてユーザに対し隙間期間中に表示するエージェント（５６、４２０）を第１のサーバ（１５）から前記メモリに通信インターフェースを用いてダウンロードしその後前記エージェントをブラウザの制御下で実行する、前記メモリが接続されたプロセッサ（３４０）とを含む、装置。

【請求項６】

埋込まれたコード（４０）を含む第１のウェブページ（３５）に応答して情報オブジェクトを、プロセッサ（３４０）と、前記プロセッサに接続されコンピュータ実行可能命令（４００）および前記第１のウェブページの両方を記憶するメモリ（３３０）と、前記プロセッサと関連して動作する出力装置（３８０）とを含むコンピュータ（５）を介して表示する際に用いる方法であって、

前記第１のウェブページは、ページコンテンツと、情報オブジェクトを選択することを求めるために埋込まれたコードとを表わす複数のコンピュータ読取可能命令を有し、前記方法は、前記プロセッサにより、前記実行可能命令に応答して、かつウェブブラウザ（７）を用いて前記コードを実行することにより行なわれる、

エージェント（４２０）を第１のサーバ（１５）から前記メモリに通信インターフェースを用いてダウンロードしその後前記エージェントをブラウザの制御下で実行するステップと、エージェントによって行なわれる、

前記コンピュータが前記第１のウェブページをユーザに対して前記出力装置を用いて表示している間に、第２のサーバによって選択された情報オブジェクトを後でプロセッサにより表示すべく用いられる少なくとも１つのファイルを通信インターフェースを用いてダウンロードするステップと、

ユーザにより発生されるクリックストリームを入力インターフェースを用いて監視することにより、第１のウェブページから次の連続するウェブページへ移行するユーザ動作を意味する、次の隙間期間の開始を意味するユーザナビゲーションイベント（５１０）を監視するステップと、

前記ユーザナビゲーションイベントに応答して、前記１つのファイルを処理することにより、前記第２のサーバによって選択された情報オブジェクトを前記出力装置を用いてユーザに対し前記隙間期間中に表示するステップとを含む、方法。

【発明の詳細な説明】

【０００１】

【発明の背景】

１．発明の分野

この発明は、たとえばインターネットのようなネットワーク化されたクライアント－サーバ環境において、クライアントコンピュータで実行中のウェブブラウザに、そのブラウザのところにいるユーザには見えない態様で広告サーバから広告がダウンロードされ、次いで、ユーザにより１つのウェブページから次のウェブページに移動すべく発生されるクリックストリームに応答してそのブラウザにより隙間ごとに表示されるネットワーク分散広告を実施するための技術、特定的には装置およびそれに伴う方法に関する。

【０００２】

２．先行技術の説明

現在、インターネットの利用および特にワールド・ワイド・ウェブ（以下単に「ウェブ」と称する）の利用は、特に、ウェブサイトおよびインターネットへのアクセスを有するユーザの数が急激に、かつほとんどが指数関数的に増大するにつれ、爆発的に増加している。

【０００３】

要するに、インターネットへの適切なネットワーク接続が確立された後、クライアントコ

ンピュータのところにいるユーザは、グラフィカルなウェブブラウザ、たとえば、ワシントン州レッドモンドのマイクロソフト・コーポレーションから現在入手可能なインターネット・エクスプローラ（「ＩＥ」）ブラウザなどを用いて、ウェブサイトに接続し次いで所望のウェブページの特定のアドレス（ＵＲＬすなわちユニフォーム・リソース・ロケータとして知られる）をブラウザに単に供給することによってそのページをダウンロードする。ＵＲＬは、そのサイトのアドレスをそのインターネットのドメイン名によって、およびそのサイトにおける情報のページをその対応のファイル名によって、それら両方を特定する。各ウェブサイトは少なくとも１つの、より頻繁にはそれよりはるかに多い数のページであって、すべて、予め規定された階層構造にて、一般にはそのルートからいわゆる「ホームページ」で始まるように配されたページを記憶する。各そのようなページはＨＴＭＬ（ハイパーテキスト・マークアップ言語）形式で書かれる。ページとは、ここにおいては、１つのＵＲＬでアクセスされるコンテンツを示し、たとえば、その特定のページに対するＨＴＭＬコードで特定されるテキスト、グラフィクスおよび他の情報を含む。ユーザが当該のＵＲＬを供給すると、そのユーザにより操作されるブラウザは適切なコマンドを、ＴＣＰ／ＩＰプロトコル（トランスミッション・コントロール・プロトコル／インターネット・プロトコル）を用いて、そのウェブサイトにありそのページを記憶する遠隔ＨＴＴＰ（ハイパーテキスト・トランスポート・プロトコル）サーバに送信することにより、そのページに対する対応のファイルにアクセスしそれをダウンロードする。これに応答して、次いで、サーバが、ＴＣＰ／ＩＰプロトコルを用いて、ＨＴＭＬコードを含む、そのページを構成する記憶されたファイルをブラウザに送り返す。そのページ自体を構成するファイルがブラウザによって受信されると、ブラウザはそのファイルの中にあるＨＴＭＬコードを解釈および実行して、そのページを適切に組立て、たとえばモニタ上において、クライアントコンピュータのところにいるユーザに対して表示する。このようなページは、それ自体が、同じまたは異なるウェブサイトにある他のファイルを参照するＨＴＭＬコマンドを含んでもよく、それらは、これらのコマンドがブラウザによって適切に解釈および実行されると、結果的にそれらのファイルはダウンロードされ、それらの結果として得られるコンテンツはブラウザによってその表示されたページに適切に組込まれる。当該ページに関連づけられたコンテンツがすべて表示されると、次いで、ユーザは、そのページ上に表示される好適なハイパーテキストリンク、ボタンまたは他の好適なユーザ入力フィールド（いずれにせよここでは「ホットリンク」を実現する）上にユーザのマウスカーソルを位置決めし、次いで、たとえばマウス「クリック」によって、ユーザがそのサイトへの訪問を終えるまで他の所望のページに対するファイルを連続して効率的にダウンロードしてそのページを連続して表示することが可能で、その訪問を終えた時点には、ユーザはホットリンクを介して他のサイトのページなどに移ることが可能である。ホットリンクは関連のページの完全なウェブアドレスを特定するもので、それは、そのページがあるそれをホスティングしているウェブウェブサイトのドメイン名を含んでいる。この結果、ユーザは、単にマウスを複数の所望のウェブページの各１つに対する適切なホットリンクに連続して位置決めし「クリック」することにより、各所望のページに対するＨＴＭＬファイルを連続してその対応のウェブサイトから容易に検索しそのページを表示し、そうすることにより、それらのサイトが物理的にどこにあるのかに関係なくサイトからサイトへ本質的に労せずしてジャンプし得る。

【０００４】

数年前におけるその導入以来、ＨＴＭＬおよびそれに伴うブラウザソフトウェアは、今ではたとえばJava言語およびJavaScript言語（「Java」はカリフォルニア州マウンテン・ビューのサン・マイクロシステムズの登録商標であり、「JavaScript」はカリフォルニア州マウンテン・ビューのネットスケープ・コミュニケーションズの商標である）のような付随のプログラミング言語を含んでいるが、それらはかなり急速で継続的な進化を経てきた。その主な目的は、ウェブページ著者に対し、彼らのページを通してますますリッチになるコンテンツを表示する能力を供給することであり、その結果、それらのページを訪問するユーザに対し「ユーザ体験」を高めることである。この結果、ウェブページは、ＨＴＭ

ＬおよびブラウザソフトウェアのＨＴＭＬの初期のバージョンで生じたような、比較的単純なテキスト形式表示にはもはや制限されず、今や、かなり洗練されたグラフィクスを使用するフルモーションマルチメディアプレゼンテーションおよび双方向型ゲームさえも包含し得る。

【０００５】

比較的低コストでインターネットにアクセスできることと結びついて簡単にウェブをブラウジングできることと、比較的容易にウェブサイトが確立され得ることが相俟って、インターネット自体、ウェブサイトおよびインターネットユーザコミュニティの世界中における他に類を見ない成長および広がりが促進されている。この点に関し、ウェブサイトを確立することにより、販売者、ベンダーおよび他の情報プロバイダは、膨大な数の潜在的顧客を－－彼らがどこに住んでいるかに関わらず－－これまで可能であると考えられたよりもはるかに少ないコストで得るという、少なくとも５～１０年前では基本的には聞いたこともないような他に類を見ない機会を有している。さらに、ウェブ上にて現在入手可能な情報の膨大な量および幅広い多様性から、ウェブブラウジングは十分な数の個人にとって非常に人気のある暇つぶしとなりつつあり、ブラウジングはかなりの視聴者をたとえばテレビおよびケーブルといった伝統的な形態のマス・エンターテインメントから離れさせつつある。このような事態は現在のところ比較的わずかなものであるが、急速に大きくなっていくように思われる。さらに、自分のパーソナルコンピュータのところにいて基本的に数回のマウスクリックより複雑なことは行なわないユーザが簡単および便利に遠隔のウェブサイトと効果的に対話し得るため、物理的な店舗を訪ねることなくインターネットを介して商品およびサービスを注文する電子商取引が大きな販売媒体として急速に台頭しつつある。この媒体は、伝統的な小売形態を大きく脅かし、おそらくは、比較的短い期間で、それらを変えさえするであろうと思われる。

【０００６】

消費者情報源としてのウェブの幅広くかつ常に成長しつつある勢力範囲および電子商取引に対する消費者の増大しつつある受容のために、広告者は、広告を消費社会に宣伝するための効果的な媒体としてウェブの大きな可能性をはっきりと認識している。

【０００７】

残念なことに、従来のウェブベースの広告は、さまざまな実際の理由から－－ある理由は性質上技術的なものであり、他の理由は伝統的なウェブ広告それら自体の性質に関連するものである－－一般的に不満足な結果を生み出しており、したがって、通常はたいていの大きな広告者からは避けられてきた。この点に関し、当該技術分野においては、ウェブベースの広告を実施するためのいくつかの方策が存在する。しかしながら、それらは、すべて、それらの所望性および使用を厳しく制限する何らかの形の厳しい制限を有する。

【０００８】

現在、「バナー」と称される、ウェブ広告用として広く用いられているフォーマットは、表示されたウェブページ内の典型的にはある固定された場所にある矩形のグラフィカルな表示の形式をとる。バナーは、静的でも動的でもよく、表示されたウェブページの任意の場所に位置し得るが、当該ページの一番上もしくは一番下または縦方向エッジに沿って位置することが多い。バナーは、そのサイズに依存して、全ページ幅または長さにわたって延在し得、通常はグラフィカルに、人目をひくような形式で宣伝対象の製品またはサービスの名称を含んでいる。ますます、所与の製品またはサービスに対するバナーは、ホットリンクを実施することにより、消費者がバナーを「クリックスルー」して（つまり、バナー上でマウスクリックを発生させて）ブラウザを介して対応の広告者により維持されるウェブサイトに移り、そのサイトからウェブページをフェッチしてその商品またはサービスに関するさらなる情報を提供する。したがって、消費者はクリックスルーによってさらなる情報を簡単に入手することが可能で、一方、広告者は、所与の時間期間で発生したそのようなクリックスルーの数を監視しながら、対応のバナーの効果に関するフィードバックを得ることができる。

【０００９】

バナーは、一般には、そのバナーが現われる所与のウェブページに対するＨＴＭＬコード

内に、そのバナーに対する適切に埋込む特定のＨＴＭＬコードを適切に埋込むことにより発生される。クライアントブラウザは、フェッチされたページに対するＨＴＭＬコードを解釈し順に実行しながら、今度は、そのバナーに対する埋込まれたコードをコンパイルおよび実行し、したがって、そのバナーを、表示されたページの一部としてその上の特定された場所に表示する。

【００１０】

バナーを実現するにあたり、静的であろうとたとえアニメート化されたものであろうと、そのＨＴＭＬ符号は一般にはそのバナーに対する適切なファイルをクライアントブラウザにダウンロードすることを伴った。このファイルは、そのページに対するＨＴＭＬファイルを記憶する同じサーバ上に記憶されていてもよく、または遠隔サーバからアクセスされてもよい。このファイルは、たとえばＧＩＦ（Graphic interchange format）のようなグラフィックそれ自体を含んでもよく、または、ブラウザによって解釈および実行されると所望の動的グラフィックを発生および表示するJavaアプレットを含んでもよい。このファイルは、それがグラフィックであれアプレットであれ、ダウンロードする時間を必要とし、ブラウザによって、ページが完全に表示される前にダウンロードされそのページ上でアセンブルされなければならない。そのファイルに対するダウンロード時間は、特にそのサイズが増大すると、明らかに、当然に、そのページが完全にダウンロードされる時間間隔を長くし、それによって、ユーザがそのページに移った後バナーを含むページを完全に表示するまでに要する時間を延ばす。モデム接続を介して与えられるような、クライアントコンピュータ（たとえばパーソナルコンピュータ（ＰＣ））へのチャネル帯域幅はかなり制限されることが多い。この結果、－－比較的「リッチな」コンテンツ、たとえば音声または映像コンテンツに対する場合では当然であろうように、バナーに対するファイルサイズが比較的大きい場合には、そのようなファイルをそのような限られた帯域幅接続を介してダウンロードする際の遅延は過度なものになり得、その結果、ユーザにとって非常に苛立たしいものとなり得る。したがって、マルチメディアコンテンツに対するすべてのページ要素が完全にダウンロードされてそのページが表示されるまでに、ユーザはかなりの時間待つことになるであろう。そのような遅延がページ移行中に生じた場合、それはユーザにとってかなり苛立たしいものとなることがあり、ユーザは、単に待機することを終わらせたいがために、そのダウンロードを途中で終わらせて他のページに移るに至り得る。したがって、ユーザに対する適切な「編集上の経験」を保とうと努力する中、コンテンツ供給者は自分のページ上に表示されるべきそのようなバナーのファイルサイズを厳しく制限することによりページダウンロードおよびしたがってレイテンシ時間を最小限にしている。

【００１１】

残念なことに、そのような制限されたファイルサイズは事実上バナーのコンテンツのリッチさをかなり単純な広告に－－動画を伴う場合でさえも－－制限する。したがって、広告者が相対的に低いクリックスルー数からすぐに認識したように、バナーはそれらの使用および費用を正当化するに十分なほどに視聴者の注意を引付けるのに失敗することが多かった。

【００１２】

バナーに関連するコンテンツ制限を克服する努力として、当該技術分野では、いわゆる「隙間」広告というさまざまな広告様式の使用が教示されている。たとえば、米国特許５，３０５，１９５（A．Ｊ．Murphyに対し１９９４年４月１９日に発行、以下「Murphy」特許と称する）には、ウェブ広告においてではないが、隙間広告を用いるという概念が開示されている。Murphy特許に記載されるとおりでは、予め記憶された広告が、特定の間隔で、ネットワーク化されたＡＴＭ（自動預け払い機）端末の群の各々上において表示される。特に、それらの広告は直接またはサーバを介して遠隔コンピュータからダウンロードされ各そのような端末上にてローカルに記憶され、次いで、遠隔のメインフレーム預け払いサーバからその端末にて開始された預け払いに対する応答を待っている間に端末上に表示される。

【００１３】
一般的に言って、特にウェブ広告に関して、隙間広告の表示は、ブラウザにより所望のウェブページを検索すべく表示されたホットリンク上にてユーザがクリックを行なってからそのブラウザがそのページを表示し始めるまでの間に生ずる時間期間中に行なわれる。そのような期間は、広くには「隙間」と称されるが、それは、ユーザが新たなページを求めてホットリンク上でクリックすると、ブラウザがその特定のページに対し遠隔ウェブサーバからファイルをフェッチし次いでそのページを完全にアセンブルおよび表示するのに時間を要するという単純な理由から生ずるものである。隙間期間の長さは全く可変であるが、それは、さまざまな要素によって支配され、それらには、たとえば、新たなページを完全に表示するのに必要とされるファイル数および各そのようなファイルのサイズ、ならびにユーザがホットリンクを活性化したときに生ずるネットワークおよびサーバの混雑および付随の遅延などが含まれる。
【００１４】
隙間ウェブ広告は、たとえば、米国特許５，７３７，６１９および５，５７２，６４３（ともに、D. H. Judsonに対し１９９８年４月７日および１９９６年１１月５日にそれぞれ発行－－以下「Judson」特許と称す）に教示されている。Judson特許には、広告を情報オブジェクトとしてウェブページファイルに埋込むという概念が開示されており、その埋込みは、ファイルが実行されてページが表示されるときに情報オブジェクトが隠されたまま表示されないような態様にて行なわれる。表示されるのではなく、情報オブジェクトはブラウザによってそのページに対するコードの実行中にローカルにキャッシュされる。次いで、ユーザによるホットリンクの活性化により開始される移行によりそのページから次の連続するページに移動する間、つまり、隙間の間に、ブラウザはその広告にローカルキャッシュからアクセスし、それを、次の連続するページがダウンロードされ表示されるような時間まで表示する。さらに、公開された国際特許出願ＷＯ　９７／０７６５６（E. Barkatらに対し１９９７年3月６日に公開）には「礼儀正しい（ポライト）」ダウンロードが教示されている。ここでは、ブラウザは、ローカルコンピュータ（たとえばクライアントＰＣ）上において、遠隔広告システムサーバから表面上はバックグラウンドプロセスとして、ウェブ広告に対するファイルを、ブラウザに接続される通信チャネル（リンク）の帯域幅利用が予め設定されたしきい値未満である期間中のみダウンロードする。そのような「礼儀正しい」ダウンロードは、そのときにそのクライアントＰＣ上で実行されている、そのリンクを利用する他の通信アプリケーションとの干渉を最小限にするよう意図されたものである。ブラウザによるダウンロードされた広告のユーザに対する表示は、ユーザが、従来のスクリーンセーバ処理によって検出されるように、予め規定された時間期間の間そのユーザのＰＣと対話しなかった後においてのみ、たとえば、その期間中にマウスを移動させたりキーボード上のキーを押したりしなかった場合にのみ行なわれる。サーバはクライアントＰＣへのダウンロードのため広告を選択するが、それは、（クライアントＰＣとサーバとの間で接続が確立されるとそのクライアントＰＣがサーバにアップロードを行なうそのＰＣのところにそのときいるユーザのユーザＩＤおよび嗜好情報と、そのＰＣの構成情報とに基づいて行なわれる。隙間広告に関連するファイルは大きなものであり得るが、これらのファイルは、クライアントブラウザによって、そのブラウザが休止ししたがってそのネットワーク接続の帯域幅利用が相対的に低くなる筈であろう期間を利用してフェッチされる。このような「休止時間」は、隙間広告の処理がない状態では、ブラウザがあるウェブページを完全に表示した後ユーザがそのページを見てはいるがホットリンク上にてクリックを行なって別のページへ移ってはいないときに生ずるであろう。このような休止時間中は、ブラウザは次のユーザ入力を待っているだけであろう。
【００１５】
バナーに固有でありかつダウンロードレイテンシによって発生される問題を除去しないにしても低減することにより、隙間ウェブ広告は、休止時間ダウンロードおよびローカルキャッシュ方式を用いることにより、非常にリッチなメディアコンテンツを満足のいく「ユーザ体験」で伝えることを理論上は約束する。しかしながら、隙間広告は、従来どおりに

実施されるとおりでは、それらの使用を厳しく制限する深刻な実質上の欠点を有する。

【００１６】

従来の隙間ウェブ広告は、他の形式の現在の－－ここではバナーとは異なる－－ウェブ広告と同様、ＨＴＭＬ広告コードをたとえば別個の表示可能でないオブジェクトとしてウェブページに対するＨＴＭＬ符号内に埋込むことに頼る。残念ながら、この方策は、Judson特許により教示されるものにおいて固有なものであるが、広告者がそれを実施し、特に、後にその広告者がどのような理由であれその広告者の広告コンテンツを変更しようと求める場合、柔軟性がなく費用がかかるものとなり得る。特に、および現在、広告符号は広告を掲げることになる各すべてのコンテンツウェブページにマニュアルで挿入されている。したがって、ますます洗練された埋込み広告、たとえばマルチメディアまたは映像もしくは音声などを既存のウェブサイトコンテンツに挿入することは、ウェブサイト、特に大きなサイトが広告に対し利用可能なコンテンツページ数を増大させるにつれ、人材、時間およびコストの面で大きな投資を必要とする。この点に関し、バナーがたとえば１行のＨＴＭＬコードの挿入を通常必要とする場合、コンテンツリッチな広告、たとえば、パラメータ化された埋込みJava広告アプレットにより現在実施されるものなどは、ページ全体の符号からなることが多く、したがって、はるかにより大規模でますます大きな労力を要しかつコストのかかる挿入を必要とする。さらに、時間が経つにつれ、広告者はその広告を－－たとえば１つの広告を全く新しいバージョンで置換えることによって－－変更する。しかしながら、一旦ＨＴＭＬ広告符号が多数のウェブページ内に埋込まれると、広告者にとって自分の広告符号が挿入された各すべてのページにアクセスし次いでその広告符号を所望のようにマニュアルで変更することは非常に非実用的であり、かなりコストがかかり得る。これらのページが他のウェブサイトにコピーされインターネット中に広まる場合、この非実用性とそれに付随するコストは一層大きくなる。

【００１７】

このような欠点がある中、当該技術においては、いわゆる「プッシュ」テクノロジを用いてウェブ広告を実施するという概念が教示されている。たとえば、米国特許５，７４０，５４９（Ｊ．Ｐ．Reillyらに対し１９９８年４月１４日に発行－－以降「Reillyら」特許と称される）を参照されたい。本質的に、およびReillyら特許に記載されるとおりでは、クライアントＰＣは「プッシュ」アプリケーションプログラム（「管理マネージャ」と呼ばれる）の実行を介して情報サーバつまり「プッシュ」ウェブサーバとのネットワーク接続を典型的には空いている時間、たとえば深夜または早朝などに、または予め規定された間隔（たとえば４時間ごと）に確立する。情報サーバは、次いで、管理マネージャに対し、たとえば広告および／または他の予め規定される情報に対する、ユーザに対しいつか後でプレイされるコンテンツファイルをダウンロードつまり「プッシュ」する。管理マネージャつまり「プッシュ」アプリケーションは、次いで、すべての「プッシュされた」コンテンツファイルをローカルハードディスク上のローカルデータベース（「情報データベース」と称される）に記憶し、情報サーバから受取られる命令に応答して、既に表示された以前に「プッシュされた」コンテンツファイルを削除する。管理マネージャはユーザプロファイルの維持も行なう。ユーザプロファイルは情報データベースにおいてユーザが受取りたい特定の広告および／または他の情報に関してユーザの嗜好を特定する。したがって、各接続を介して、情報サーバは、ユーザプロファイルで特定される嗜好に対してそのデータベースからコンテンツを選択することにより、クライアントＰＣに対し、ユーザにとって興味の対象であろうフレッシュなコンテンツを、既に表示されたものと重複することなく「プッシュ」しようと試みる。記憶された「プッシュされた」コンテンツは、後で、データビュワーを用いながら、ユーザの要求に応じて、または、ここでも従来のスクリーンセーバ手順にて検出される、ユーザがシステムと対話中でないときに表示される。

【００１８】

プッシュテクノロジはダウンロードを空いている時間に生じさせるようシフトすることによりダウンロードレイテンシを減ずる一方で、この技術もその実用的な受入れを大きく制限する深刻な欠点を有している。

【００１９】
特に、「プッシュされた」コンテンツにアクセスするためには、ユーザは、まず、自分のクライアントＰＣに対し、プラットフォームに対し特化されたソフトウェアアプリケーションプログラムのダウンロードおよびインストールと、そのプログラムの製造者がリリースする新たなプッシュ能力に伴う当該プログラムに対する後の更新のダウンロードおよびインストールとを行なわなければならない。残念なことに、これらのアプリケーションプログラムは長さが数１０メガバイトに及ぶことが多い。典型的なインターネットユーザは自分のインターネットサービスプロバイダにモデム接続を確立するので、これらのユーザはこれらの相対的に大きなプログラムファイルをダウンロードすることはたとえ圧縮された形式であっても時間が非常にかかり、ユーザが積極的に自分のクライアントＰＣを用いている間は一般的には非実用的であることを知るだろう。この結果、これらのユーザは、いくらかのコストをかけてそのアプリケーションプログラムの在庫バージョンを購入するか、または、そのプログラムを、典型的にはそのプログラム自体に対してはコストをかけずに、空いている時間に、ネットワークの混雑が比較的少ないときにダウンロードすることを余儀なくされる。さらに、当該分野では、増分的なソフトウェア更新をクライアントＰＣに自動的に「プッシュ」しおよびインストールすることによりユーザがそれをマニュアルで行なう必要性を取除く努力が行なわれているが、それでもユーザは依然として「プッシュ」アプリケーションプログラムの初期ダウンロードおよびインストールに伴う負担を負う。

【００２０】
加えて、「プッシュ」アプリケーションプログラムは、追加される能力をユーザンに与えるため、サイズが大きくなり続けており、それはかなり大きくなることが多い。プッシュアプリケーションをダウンロードし次いでそれを定期的に更新することは、ユーザが自分のクライアントＰＣにおいて利用可能なディスク空き領域量を時としてかなり低減することになる。さらに、「プッシュ」アプリケーションは、周期的に大量のメディアコンテンツをプッシュサーバからクライアントＰＣに「プッシュ」しそのコンテンツを後の表示を保留してＰＣのハードディスクに記憶することに依存する。このコンテンツは、そのボリュームによって、あまりに大きなハードディスク空き領域を占め得る。さらに、広告者は、意外なことではないが、比較的少数のＰＣユーザしか、たとえばアプリケーションプログラムをダウンロードしインストールすることなどによる積極的な行動を－－そのサイズにはほとんど関係なく－－とって広告および他のそのような情報を受けようとはしないことも知った。

【００２１】
従来提供されてきた隙間または「プッシュ」によるウェブ広告に伴うこれら実際上の、およびかなり深刻な欠点に直面して、ウェブ広告者は、彼らの広告をバナーのような方策で広告ＨＴＭＬファイルの実時間ダウンロードおよび表示を介して表示することにその努力を向けたようであった。ここでは、広告ファイルは、所与のウェブページＨＴＭＬファイル内に埋込まれるのではなく、遠隔のウェブサーバ上にあり、適切なＨＴＭＬタグ、これは広告ファイルを参照するのであるが、それがウェブページファイル自体に埋込まれる状態で伴われている。このようなタグによって、そのページ内において広告がいつおよびどこに現われるかが特定される。

【００２２】
このようなバナーのような広告において本質的なレイテンシ問題を克服するため、さまざまな各社固有のメディアフォーマットが当該技術分野において登場した。これらのフォーマットは、ますます洗練されるデータ圧縮を、ときに映像および／または音声ストリーミングとの関連で用いる。メディアファイルがそれが表示される前に完全にダウンロードするのを待つのではなく、ストリーミングは、「ストリーム化された」メディアファイルにおけるコンテンツがユーザのクライアントブラウザに到着するとそのコンテンツをリアルタイムでユーザに呈示する。この方策は明らかに従来のバナーを介して得られ得るよりもコンテンツのリッチさが向上し、したがって「ユーザ体験」を高め得る一方で、それにも

かかわらず、遠隔のウェブサーバへのリアルタイムのネットワーク接続が継続的に存在していることに依存するという不利な点を持つ。

【００２３】

残念なことに、ダウンロードを停止するどのようなネットワークまたはサーバ混雑も、たとえそれが一時的なものであっても、「ストリーム化された」メディアプレゼンテーションをユーザに対しそれが完了する前に一時停止つまりフリーズまたは全くフォールトし得る。この中断は、それが気付き得るものであり十分長い場合には、ユーザを苛立たせ「ユーザ体験」を損なうであろう。

【００２４】

これらの欠点にもかかわらず、特に隙間広告およびプッシュテクノロジに関し、および明らかによりよい代替策の欠如から、現在使用中の大半のウェブ広告は、グラフィックファイルのリアルタイムストリーミングを、それらのコンテンツがブラウザによって表示される状態で用いている。

【００２５】

ウェブ広告は他の形式の大量広告と同様収入をもたらすが、それは、この場合はウェブサイト所有者である広告主に対する継続する支払いの流れの形態をとることが多い。広告が実際に広められる程度を仮定して広告者が過剰または過小請求されないことを保証するためには正確なユーザ決算が重要である。したがって、これらの支払いは、広告に達したウェブユーザ数の関数と結び付けられることが多い。しかしながら、ウェブ広告の場合、その数を正確に確かめることは困難でよくいっても問題があり、そうするために用いられる基本的な技術を仮定しても、明らかにエラーが生じがちであり、したがって、信頼性のないユーザ数および誤った広告料金をもたらす。

【００２６】

特に、および従来どおりに利用されるとおりでは、たとえばストリーム化された広告のようなウェブ広告の配信は、「ユーザ到達」として、ウェブサーバにおいて、広告ファイルたとえばストリーム化されたファイルが供給される瞬間にログされ、ブラウザがその広告をユーザに完全に表示した後には行なわれない。残念なことに、これらの広告ファイルを供給することは、これらのファイルが最終的および完全にクライアントブラウザによってユーザに表示されることを保証しない。この結果、ウェブサーバにより発生される「ユーザ到達」数はひどく誇張されるかまたは控えめに示される。たとえば、ユーザが新しいコンテンツページへのナビゲーションを広告のプレイ開始後であってかつその広告の完成前において行ない、そうすることによってその広告を途中で終了させた場合でも、ある完全な到達がそれにも関わらず――誤って――ログされる、というのも、その広告は完全に供給されたからである。代理サーバが、イントラネットまたはローカルエリアネットワーク（ＬＡＮ）上にある複数のクライアントＰＣと、インターネット（または他の不確かな公共ネットワーク）上にあるウェブ広告サーバとの間にある場合には、さらなるエラーが生ずる。この場合、広告ファイルに対するクライアントＰＣのうちの１つからの要求が代理サーバに経路制御されるとその代理サーバはその要求を広告ウェブサーバにさらに向ける。後者は、その要求に応答して、広告ファイルの完全なコピー１部を代理サーバに供給する。その結果のフェッチされた広告ファイルは代理サーバでローカルにキャッシングされ、そこから、要求を行なっているクライアントＰＣに与えられる。他のクライアントＰＣのいずれかが同じファイルを要求する場合には、代理サーバはこれらのファイルを、ウェブサーバには全く知られずに、そのローカルキャッシュから与え、当該の他のＰＣからの要求をウェブサーバに向けて返しはしない。したがって、ウェブサーバは、代理サーバが広告ファイルに対しそのローカルキャッシュからアクセスしてその広告を最初に要求したクライアントＰＣ以外の任意のクライアントＰＣに広めたというさらなる事実には全く気づかない。代理サーバの背後にあるいくつかのイントラネットは数万または数十万の個々のクライアントＰＣを伴ってかなり広範囲となり得るため、ウェブサーバによって配信されるコピーに基づくサーバベースのユーザ到達数決算は、代理サーバの存在のため、あまりに低いものとなって、広告者に対する非常な過小請求をもたらすかもしれない。当該技

術分野において、ウェブ広告の「ユーザ到達」の正確な数を与え得る解決策は未だ存在しないようである。

【００２７】

相対的に低速な通信リンクたとえばモデム接続などを介してコンテンツファイルをダウンロードすることに伴うレイテンシ時間を低減することを目指した他の従来の方策は、さまざまなプログラミング言語内における新たな機能の開発および使用を伴った。これらの方策は、最も顕著にはJavaおよびJavaScriptプログラミング言語を伴い、有用である一方で、利用可能なリンク帯域幅の使用が依然として不十分であり、コンテンツファイルのサイズを依然として制限する。これらの制限は、ユーザが新しいウェブページに移るごとに、予めロードされたファイルを途中で終了させることから生ずる。具体的には、これらの方策では、広告ファイルのダウンロード中でそのダウンロードが完了する前にユーザがホットリンクを活性化して新しいウェブページに移る場合、そのダウンロードは単に停止する。そのダウンロードは、次にその特定の広告ファイルが要求されるときに再開されることを必要とするが、それは、そのファイルの初めから行なわれる。したがって、その広告ファイルの一部をダウンロードする際にそのとき費やされた時間および帯域幅は完全に無駄になる。実際、多くのユーザは、所望の目的地に到達するまで一連のウェブページを通って素早くナビゲートする傾向にある。その結果、広告者は、単位時間あたりの途中での終了の数を減じ、そうすることによって、その同じ広告ファイルの重複部分をダウンロードすることにより生ずるレイテンシを減じようとする努力の中、やはり、コンテンツファイルのサイズおよびしたがって自分たちの広告の「リッチさ」を最低限に抑えることを余儀なくされる。したがって、これらの方策は一般に全く不満足なものであることがわかっている。

【００２８】

当該技術分野で公知であるさまざまなウェブベースの広告技術に伴う根本的な欠点に鑑みると、隙間ウェブ広告はこれらすべての技術に対する最も大きな可能性を有するように思われる。しかしながら、従来の隙間広告の従来的実現例に内在する制限により、事実上、この形式のウェブ広告はその可能性を実際には達成していない。さらに、すべての公知のウェブ広告技術に内在する欠点は、全体として一般にウェブ広告の使用をかなりな程度に禁じていた。

【００２９】

このように、当該技術分野においては、その技術分野において公知の隙間広告技術に伴う弱さを有さない新しいウェブベースの隙間広告技術が切に求められている。

【００３０】

この点に関し、この新しい技術は、好ましくは、広告ＨＴＭＬファイルをウェブページ内に埋込むべきではない。これが達成され得る場合、有利なことにそのような技術は、広告者に対し、ウェブページファイルへの広告の挿入およびそれらのうちいずれかの広告の後における変更の両方に関し、節約された労力、時間およびコストで、かなりの経済効果を与えるであろう。加えて、そのような新しい技術は、全くではないとしてもほぼユーザに見えない形であってそのユーザにとって不便でも負担でもない態様で好ましくは機能すべきである。特に、好ましくは、この新しい技術は、ユーザが自分のＰＣに別途のアプリケーションプログラムのダウンロードおよびインストール、ましてやその何らかの更新を行なうことにより特にウェブ広告を受信する必要もなければ、通常のウェブブラウジング以外の何らかの積極的な行動を起こしてそのような広告を受信する必要もなくあるべきである。さらに、好ましくは、この新しい技術は、プラットフォーム非依存型であり、そうすることによって、実質的に任意のＰＣ上の実質的に任意のウェブブラウザとともに動作すべきである。さらに、この新しい技術は、使用に際して、好ましくは、クライアントＰＣ上の過剰なハードディスク空き領域を占めるべきではない。さらに、満足のいく「ユーザ体験」を与えるために、この新しい技術は、広告の表示を、完全に、かつ、他の態様ではネットワークおよび／またはサーバ混雑の結果生ずるかもしれないどのような中断もなく行なうべきである。最後に、この新しい技術は、完全に表示された広告のユーザ到達を正

確かつ有効に確認することにより、広告者に対し適切な決算を与えるべきである。
【００３１】
このような新しいウェブベースの隙間広告技術が提供されれば、その技術は、効果的でありかつ望ましいものであるはずであるから、広告者による幅広い支持および使用ならびにウェブユーザによる受入れを十分に達成し、したがって、ウェブベースの広告の使用を一般に実質的に拡大するであろう。
【００３２】
【発明の概要】
有利なことに、我々のこの発明的な技術は従来のウェブベースの隙間広告技術に関連する欠点を克服することにより上記の要求を満たす。
【００３３】
我々のこの発明はこれを、我々の広範囲にわたる発明的技術に従って：広告コンテンツをウェブコンテンツページ（これ以降「参照」ページとも称する）から「結合を解除」し；広告ファイルを、クライアントコンピュータで実行されるブラウザを介して、そのコンピュータにあるブラウザキャッシュ（たとえばブラウザディスクおよびＲＡＭキャッシュ）に、そのブラウザのところにいるユーザには見えない態様で「礼儀正しく」ダウンロードし；異なるウェブページにわたる通常のユーザのナビゲーションに関連するユーザのクリックストリームに応答して広告をブラウザを介して隙間に表示することにより達成する。
【００３４】
具体的には、我々の技術はＨＴＭＬタグ（必要な場合には、他のＨＴＭＬタグと区別するため、これ以降「広告タグ」とも称される）を参照ページに埋込むことに依存する。このタグは２つの要素を含む。一方の要素は実際に配布ＨＴＴＰ（ウェブ）サーバから必要な程度にダウンロードを行ない、次いで、クライアントブラウザにて、「軽量」Javaアプレットとして実施されるエージェントのインスタンスの生成を持続して行なう。このエージェントは、次いで、第３者の広告ＨＴＴＰ（ウェブ）サーバ上にある広告管理システムから発生される、所与の広告に対する広告ファイル（メディアファイルおよび必要な場合にはプレイヤーファイル）をブラウザディスクキャッシュ（さらにメディアファイルの場合にはブラウザＲＡＭキャッシュ）に「礼儀正しく」および見えない形でダウンロードし、その後、それらのメディアファイルをブラウザを介して隙間ごとにかつユーザのクリックストリームに応答してプレイする。他方の要素は、広告ファイルがそこからダウンロードされるべき広告管理システムの、ウェブアドレスに関するリファレンスである。この後者のリファレンスは広告コンテンツをウェブページから全く「結合を解除」するが、それは、ウェブページが、従来生ずるように実際の広告コンテンツをそのページそのもの内に埋込むのではなく、ＵＲＬを介して特定の広告管理システムを参照し、特定の広告やそのコンテンツは参照しない広告タグを単に含む。広告管理システムが、ダウンロードされるべき所与の広告を選択するのであり、そのセレクションおよびそのコンテンツがウェブコンテンツページに埋込まれるのではない。
【００３５】
有利なことに、エージェントは、クライアントブラウザにおいて、どのような参照ウェブページ内のコンテンツからも独立して動作する。一旦ロードされ開始されると、エージェントは並列に、標準的なブラウザ機能性を伴って実行され、継続的および目に見えない形で広告を要求しそれをクライアントコンピュータ（たとえばパソコン－－ＰＣ）にあるブラウザキャッシュにダウンロードしてそれらの広告を隙間にプレイする。
【００３６】
特に、一旦エージェントが開始されると、そのエージェントは、広告管理サーバから発生される、広告に対するメディアファイルおよびプレイヤーファイルであって、その広告においてコンテンツを完全にプレイすることを必要とされるメディアファイルおよびプレイヤーファイルの両方をクライアントブラウザを介してブラウザキャッシュに礼儀正しくかつ見えない形でダウンロードする。さらに、エージェントは、そのときブラウザを動作させるユーザにより発生されるクリックストリームも監視する。クライアントブラウザに対

し次の連続するコンテンツウェブページに移るよう命じかつ隙間期間の開始を意味するたとえばマウスクリックなどのユーザにより開始されるアクションに応答して、エージェントは、すべてのメディアファイルおよびプレイヤーファイルがそのときクライアントハードディスク上にある場合は、メディアファイルをブラウザを介してその隙間期間中にそのブラウザキャッシュから直接プレイする。広告は隙間で典型的にはそれらがクライアントブラウザにダウンロードされた順序でプレイされる。ブラウザキャッシュからの隙間プレイは、有利なことに、以前にキャッシュされたコンテンツリッチな広告が、ブラウザを介して、クライアントブラウザにそのとき利用可能である通信リンク帯域幅に悪影響を及ぼすことなくプレイされることを可能にする。したがって、広告がプレイされている最中に、十分な利用可能なリンク帯域幅を用いて次の連続するコンテンツウェブページをダウンロードし得る。

【００３７】

キャッシュされた広告のプレイを引起すユーザのクリックストリームを用いることにより、ユーザは、広告を受信するのに、通常のウェブブラウジング以外の何らかの積極的な行動をとる必要もなければ、どのような新しい手順を学ぶ必要もなく、したがって、ユーザに対しさらなる負担をかけないという利点がある。

【００３８】

有利なことに、エージェントは、広告サーバから発生される広告メディアファイルおよびプレイヤーファイルを、ブラウザキャッシュに対し、他の態様ではブラウザ休止時間となるであろう間、つまり、ウェブページがユーザに対して表示されている最中でありブラウザがユーザ入力に対し待機している間に、「礼儀正しく」ダウンロードする。この態様で広告ファイルをキャッシュすると、有利なことに、ウェブ上で配信される従来のストリーミングされかつ静止しているメディアでは頻繁に生ずるさまざまなレイテンシおよび不安定な（たとえば間欠的または一時停止される）プレイが避けられる。

【００３９】

隙間期間の開始で、エージェントは、所与の広告（典型的にはそのいわゆるAdDescriptor（広告記述子）ファイルがプレイ待ち行列の先頭にあるもの）をプレイするよう要求されるメディアファイルおよびプレイヤーファイルがすべてそのときクライアントＰＣのディスク上にあるか、または、メディアファイルに関してはブラウザＲＡＭキャッシュにあるかどうかを判断する。そうである場合、次いで、エージェントは、その広告を「プレイ」すべくそのディスクからこれらのファイルにアクセスする。すべてのメディアファイルおよびプレイヤーファイルはそのときローカルに存在するので、広告は、ユーザの視点からは、即座にクライアントハードディスクまたはブラウザＲＡＭキャッシュから本質的に全くダウンロード遅延を伴わずに表示され、したがって、非常に満足のいく「ユーザ体験」を、現在のＣＤ－ＲＯＭベースの配信を介して得ることができるコンテンツに近いリッチなマルチメディアコンテンツを伴って与える。この後、エージェントは制御をブラウザに返して、ブラウザは、次の連続するウェブページがダウンロードおよびアセンブルされて表示される準備がある場合には、その特定のページをユーザに表示する。しかしながら、広告がユーザによって途中で終了される場合には、その広告は（そのAdDescriptorファイルに関し）プレイ待ち行列に（そのメディアファイルおよびプレイヤーファイルがクライアントハードディスク上に、またはメディアファイルの場合にはブラウザＲＡＭキャッシュに残る状態で）残り、次の連続する隙間期間の開始でその初めから再生されることになる。さらに、広告に対するメディアファイルおよびプレイヤーファイルのダウンロードがユーザのクリックストリームつまり隙間期間の開始によって仮に中断される場合、エージェントは、後に続く隙間期間が終了する後までさらなるダウンロードを一時停止する。通信リンク帯域幅を節約するために、エージェントは、次いで、これらファイルのダウンロードを、それが一時停止された点から再開し、従来生ずるようにダウンロードを全体的に再開することはしない。

【００４０】

我々の具体的な発明的教示に従うと、エージェントはTransition Sensor（移行センサ）

アプレットと「AdController（広告コントローラ）」アプレットとの２つのアプレットを含む。Transition Sensorアプレットのみそれ自体が任意のコンテンツページに関連づけられる。AdControllerアプレットは、一旦開始されると、ブラウザの下で実行されるが、それはブラウザそれ自身の制御下にはない。

【００４１】

広告タグはそれ自体がコンテンツウェブページに埋込まれる。広告タグは、その要素の１つとして、配布サーバに記憶されるJavaScriptファイル（「スクリプト」を含む）を参照する。JavaScriptファイルは、実行されると、アプレットタグの動的書込を介して、Transition Sensorアプレットを実行、ダウンロードおよび実施する。この特定のアプレットは、自分のブラウザでそのページに対するＨＴＭＬコードを表示するユーザの目には見えないままである。特に、JavaScriptファイルがダウンロードされ、それが含むスクリプトが次いでブラウザによって実行されると、そのスクリプトは、予め規定された数および組合せのアプレットタグ、つまりともになってTransition Sensorアプレットを形成するものを、広告タグの代わりに、検索されたウェブページコンテンツに動的に書込む。これらのタグが後にクライアントブラウザによって実行されると、Transition Sensorアプレットが呼出される。上で論じたように、広告タグも、その要素の他方として、典型的には、具体的なメディアおよびそれに付随するプレイヤーファイルを含む、全体でウェブ広告を構成する第三者の広告サーバ上にある特定の広告管理サーバへのリファレンスつまりＵＲＬをカプセル化する。

【００４２】

特に、実行されるとTransition Sensorアプレットは、アプレット間通信用に用いられるApplet Registry（アプレットレジストリ）のインスタンスを生成する。この後、Transition Sensorアプレットによって、AdControllerアプレットがブラウザディスクキャッシュにダウンロードされたかどうか、またはこの特定のアプレットの更新されたバージョンが配布サーバ上にあるかどうかが判断される。このアプレットの更新されたバージョンがブラウザディスクキャッシュに以前にダウンロードされたものに対して配布サーバ上に存在する場合、またはこのアプレットがこのキャッシュ上に全くダウンロードされなかった場合には、Transition SensorアプレットはAdControllerアプレットを配布サーバからブラウザディスクキャッシュにロードする。Transition Sensorアプレットは、次いで、AdControllerアプレットのインスタンスを生成する。一旦これが生ずると、Transition Sensorアプレットは、次いで、それ自身およびAdControllerアプレットに対しApplet Registryにおいて適切なエントリを確立する。

【００４３】

Transition Sensorアプレットは次いで、広告管理システムのＵＲＬを、広告タグに特定されるとおりに、AdControllerアプレットに送ることにより、後者のアプレットが、そのシステムから発生される広告、具体的には関連づけられるAdDescriptorファイルの配信を要求するようにする。次いで、システムは、配信されるべき広告を選択し、第３者広告サーバを介してその情報を、AdControllerアプレットに対し、要求されたAdDescriptorファイルを返すことにより伝える。所与の広告に対し、この特定のファイルは、本質的にはテキスト形式であるが、明確な、つまりファイル名およびその広告に対するコンテンツを構成するすべてのメディアファイルおよびそれらメディアファイルのすべてをプレイするのに必要なすべてのプレイヤーファイルの対応ウェブアドレスと；それらさまざまなメディアファイルがプレイされるべき順序と；各プレイヤーが対応のメディアファイルを適切にプレイするよう各プレイヤーの動作を構成および動作させるのに必要とされるさまざまな構成および他のパラメータとからなるリストを含む。AdControllerは、次いで、「礼儀正しく」、典型的には広告配布サーバを介して、関連のメディアファイルおよびプレイヤーファイルを、AdDescriptorファイルに特定される通りに－－およびそれらがクライアントＰＣのハードディスク上に既に存在しているのでない限りにおいて－－ダウンロードする。上に記したように、Transition Sensorアプレットは、さらに、現在のユーザにより発生されるクリックストリームを監視することにより、ユーザにより開始されるページ移行

およびしたがって隙間期間の開始を検出する。

【００４４】

有利なことに、AdDescriptorファイルは、メディアファイルおよびプレイヤーファイルを参照ウェブページから全く分離ししたがって広告コンテンツそれ自体がそのプレゼンテーションを呼出したコンテンツウェブページから完全に独立したままであることを保証するデータ抽象化を実施する。この抽象化によって、我々の技術は、リッチなウェブ広告、特にメディアファイルおよびプレイヤーの複雑な集合を必要とするものを配信するための非常に効果的な、汎用的かつ非常に柔軟性のある機構を提供する。この抽象化を用いることにより、我々の技術は、現在および将来のメディアフォーマットを、それらの要件に関係なく、配信機構としてJavaアプレットに依存する各社固有のストリーミングおよび他のコンテンツ配信技術を含んで－－すべてユーザに見えない形で－－対応することができる。さらに、AdDescriptorファイルはメディアファイルおよびプレイヤーファイルをそのとき使用されている異なるブラウザ、オペレーティングシステムおよび計算プラットフォームに対し特定化し得るので、我々の技術は幅広い多様な異なる計算プラットフォームおよびブラウジングプラットフォームを伴って容易に機能し得る。

【００４５】

Transition SensorアプレットおよびAdControllerアプレットは、各々、適切なJavaクラスを介して実装され、ブラウザディスク内における記憶を通し、あるサイト内における異なるコンテンツページ、異なるウェブサイト、および連続するブラウザセッションにわたってまとまって存続する。一旦これらアプレットのいずれかが完全にダウンロードされると、ユーザがそのウェブ上においてウェブサイトをわたってナビゲートする際にそれが続いてブラウザディスクキャッシュからフラッシュされない場合には、そのアプレットに対するファイルは、そのアプレットが必要とされる次の機会、たとえば、ユーザが次に現在のブラウザセッションまたは後のセッション中に、広告タグを含む任意のコンテンツページにナビゲートするときなどに、そのキャッシュからロードされ、配布サーバからのダウンロードはされない。

【００４６】

クライアントブラウザが、広告タグを含む連続する次のページに遭遇するたびに、そのブラウザはまず自動的に配布サーバに問合せをしてTransition Sensorアプレットに対する実行可能なコードが以前にブラウザディスクキャッシュにダウンロードされている場合にはそれが更新されたバージョンによって上書きされなかったことを確認する。そのような更新されたバージョンが存在する場合には、ブラウザは更新されたファイルを配布サーバからまとめてダウンロードして、そのブラウザディスクキャッシュ内にある各Transition Sensorアプレットファイルをその更新されたバージョンと必要な程度にまで置換える。代替的に、Transition Sensorアプレットが以前にブラウザディスクキャッシュにダウンロードされていなかった場合には、ブラウザはTransition Sensorアプレットに対するすべての必要なファイルを配布サーバからそのキャッシュにダウンロードする。Transition Sensorアプレットは、一旦実行されると、ブラウザを介してAdControllerアプレットをロードする。そうするために、ブラウザは、必要な場合には、更新されたバージョンを、配布サーバから、それがTransition Sensorに対して行なったのと同じ態様で得る。この結果、エージェント（具体的にはTransition Sensorおよび／またはAdControllerアプレット）がクライアントブラウザに最後にダウンロードされてからそのエージェントに対してなされたどのような訂正または改善も、ユーザの視点から見てそのブラウザに自動的かつ目に見えない形で配布され、次にブラウザが広告タグを含むウェブページに遭遇したときにブラウザディスクキャッシュにダウンロードされる。この態様で動作させることにより、ユーザは、アプリケーションプログラムをロードおよびインストールして広告を得ることおよび／または後にそのプログラムを更新することの両方に対するどのような必要性からも全くかつ有利に解放される。

【００４７】

さらに、エージェントは、有利なことに、バックグラウンドにおいて目に見えない形で、

ユーザによる共通ウェブサイト上のページおよびウェブサイトにわたってのナビゲーションから独立しかつ見えないように存続および機能する。エージェントは、実際には、クライアントブラウザにより実施される通常のＨＴＭＬおよびＨＴＴＰ処理と並列かつそれに見えない形で実行されるバックグラウンドプロセスを実施する。

【００４８】

さらに、ウェブ広告の従来のサーバベースの決算とは大きく対照的に、我々の発明的技術は各ユーザ到達の非常に正確なクライアント側決算を可能にする。各ログエントリは、AdControllerアプレットによって発生され、クライアントブラウザでのある完全な広告の成功裡なプレゼンテーションを特定する。このエントリは、広告コンテンツのソース、つまり、関連の広告管理システムのＵＲＬに関しその広告のタイトルおよび参照ウェブページのＵＲＬを含んでもよい。他のクライアント側情報は、各エントリにおいて測定され含まれ得るが、たとえば：広告がブラウザにより表示される間（恐らくはユーザがその広告上にいる間）の時間量；ならびに、ＵＲＬに関し、ユーザが次にナビゲートしたコンテンツウェブページの識別情報（特にユーザが、そのページに、その広告に表示されるホットリンクを介して到達した場合）などである。これに次いで、AdControllerアプレットはログエントリを広告サーバにアップロードする。これらのエントリは必要に応じてまとめて処理されることにより、ウェブベースの広告者からの共有される広告収入が異なるウェブページコンテンツプロバイダ間で適切に分配される。

【００４９】

有利なことに、我々の発明的技術は、参照ウェブページコンテンツをその対応の広告コンテンツから全く結合を解除することにより、容易に、広告者が、その広告のうち任意のものの変更または更新を、単に、第三者広告管理システムにある適切なメディアファイルまたはAdDescriptorファイルを必要に応じて修正することにより行なうことを可能にする。参照ウェブページは、単に、広告コンテンツを全く欠く広告タグを組み入れるにすぎないので、何の変更もそのページに対してなされる必要はない。したがって、我々の発明的な技術を用いれば、従来的に必要とされるものに比して、ウェブベースの広告の維持および更新に伴う負担、時間およびコストが実質的に低減される。

【００５０】

ある特徴として、我々の発明的な技術は、有利なことに、その持続するエージェント方策と組合わされて、マルチスレッド化されたパイプライニングを実施する。各異なる広告を異なるスレッドとして処理することにより、異なる処理動作のシーケンスの各々が、実際上パイプライン化されたパラレルベースで、異なるシーケンス状に生ずる広告上において実行され得、したがって、広告がプレイバックに対し待ち行列に入れられ得る率を向上させ（スループットを増大させ）得る。加えて、このようなパイプライニングを通して、完全に呈示された広告のロギングがパイプラインにおいて最後の動作としてかつ本質的に：そのAdDescriptorファイルがちょうど呈示された広告に対するもののすぐ後ろのプレイ待ち行列にあるキャッシュされた広告のプレゼンテーション、または次の連続する広告のダウンロードおよびキャッシュのいずれかと並列に生じ得る。

【００５１】

この発明の教示は添付の図面と併せて以下の詳細な説明を考慮することにより容易に理解され得る。

【００５２】

理解を容易にするため、可能な場合には、同一の参照番号を用いることにより、図面に共通の同一要素を示した。

【００５３】

【詳細な説明】

以下の記載を検討すれば、当業者には、我々のこの発明の教示が、広告または他の情報がユーザに対し隙間期間中つまり連続して表示されるウェブページ間における移行中に呈示される任意のネットワーク化されたクライアント－サーバ環境にて利用され得ることがはっきりと認識されるであろう。そのような環境には、インターネットもしくはイントラネ

ット、またはクライアントブラウザ（そのブラウザが専用クライアントコンピュータ上にて実行するか否かにかかわらず）を用いてウェブページまたはより一般的にはファイルにネットワーク通信チャネル（リンク）を介してアクセスしサーバ（やはりそのサーバが専用コンピュータ上にて実行するか否かにかかわらず）からダウンロードするような任意のクライアント－サーバ環境が含まれる。この点に関し、サーバは、そのネットワーク化された環境においてそのコンピュータ上にて実行する別個のソフトウェアアプリケーションでよく、そのコンピュータがそれ自身、ネットワークにおける他のサーバへのクライアントであってもよい。

【００５４】

簡単に説明するため、および読み手の理解を促すため、我々は、我々のこの発明を、例示的な使用の流れで、隙間の、ウェブベースの広告をインターネットに接続されるクライアントのパソコン（ＰＣ）に表示し、より具体的には、そのＰＣにおいて実行されるクライアントブラウザを用いてウェブページを遠隔のネットワーク化されたインターネットアクセス可能ウェブサーバからダウンロードし表示することにおいて論ずる。明らかに、以下の説明を検討すれば、当業者ならば、我々の発明の教示が、クライアントが、ネットワークに接続される特定のコンピュータのみならず、ソフトウェア処理であって、他のそのような処理へのネットワーク接続性を処理して情報を要求し、応じて、後者により供給される情報を得るようなソフトウェア処理をも含み得るような任意のクライアント－サーバ環境または同様の分散処理環境に、どのように容易に組み入れられ得るかを容易に理解するであろう。

【００５５】

我々は、まず、我々の発明の概観を、特に、クライアントＰＣにおけるインターネットウェブブラウザとともに用いる流れで論じ、それに続いて、その実現例の各基本構成要素について記載する。

【００５６】

Ａ．概観

我々の発明のインターネット環境におけるおおよその展開は、図１Ｂおよび図１Ｃにおいてまとめて示されているが、これらの図に示されるプロセッサ間エージェントダウンロード／インスタンス生成動作５０のうちある部分の詳細な図は図１Ｅおよび１Ｆに示されている。図１Ｂおよび１Ｃならびに１Ｅおよび１Ｆに対する図面の正しい並び方は図１Ａおよび１Ｄにそれぞれ示される。図２Ａおよび２Ｂは、それらの図に対する図面の正しい並び方が図２に示されているが、まとまった状態で、我々の発明を目に見えない形で呼出す汎用化されたウェブページＨＴＭＬコード、およびウェブ広告をダウンロードし表示するために我々の発明がそのコードに対してダイナミックに行なう変更を示す。理解のため、以下の説明を通して、図１Ｂおよび１Ｃ、１Ｅおよび１Ｆ、ならびに２Ａおよび２Ｂを同時に参照されたい。

【００５７】

図示されるように、クライアントＰＣ５は、その上でクライアントブラウザ７が実行されるが、通信リンク９を介してインターネット１０に接続される。ブラウザ７は、マイクロソフト・コーポレイションまたはネットスケープ・コーポレイションからそれぞれ市場で入手可能なインターネット・エクスプローラまたはネットスケープ・ナビゲータなどのような従来のウェブブラウザである。好ましくは、後ですぐ明らかになるが、そのブラウザは、好ましくは、アプレットタグの動的書込をサポートすべきである。コンピュータ間動作を簡単に説明するため、我々はインターネット１０を部分10_Aおよび10_Bを有するものとして図示しているが、我々はそれら両方の部分を単にインターネット１０とまとめて呼ぶことにする。ウェブサーバ１３はリンク１１を介してインターネット１０に接続されており、任意のウェブＨＴＴＰ（ハイパーテキスト・トランスファ・プロトコル）サーバを表わす。このサーバは、ウェブブラウザ７からの特定のファイルをフェッチする要求に応答して、そのファイルを、従来のＴＣＰ／ＩＰプロトコル（トランスミッション・コントロール・プロトコル／インターネット・プロトコル）を用いて、インターネットを介し

てブラウザ７にダウンロードする。ブラウザ７は、次いで、そのファイルを、典型的にはモニタ上において、そのクライアントＰＣのところにいるユーザに対して表示する。

【００５８】

広告配布ＨＴＴＰサーバ（「エージェント」サーバとも称される）１５は通信リンク１７を介してインターネット１０に接続され、予め規定されるエージェント、具体的には軽量Javaアプレットをまとめて実施するファイルを記憶する。このエージェント（ここにおいては「AdController」エージェントと称される）はそれ自身を、メディアリッチな広告コンテンツと並んで、クライアントＰＣ５上のブラウザに関連づけられるローカルハードディスクキャッシュ（「ブラウザディスクキャッシュ」）に目に見えない形で予めロードする。サーバ１５はAdControllerエージェントを以下に記載される態様でクライアントブラウザ７にダウンロードする。このエージェントは、一旦インスタンスを生成されスタートされると、次いで、目に見えない形でかつ礼儀正しく広告をブラウザディスクキャッシュにダウンロード（実際には予めロード）し、次いで、それら広告の各々のプレイを、隙間ごとに、ユーザがブラウザ７を用いて連続するウェブページ間をナビゲートする際にユーザが発生させるクリックストリームに応答して行なう。このようなハードディスクキャッシュ方式は、有利なことに、インターネットを介して配信される従来のストリーム化された静的メディアに伴う可変のレイテンシおよび不確かなプレイを回避する。このエージェントはリッチな広告が非常に制御された態様にて呈示されることを可能にし、その結果、ユーザ体験はＣＤ－ＲＯＭのそれに近づく。

【００５９】

第三者広告ＨＴＴＰサーバ２０は、インターネット１０にたとえば通信リンク１８および２３を介して接続されており、管理システム２５のホストである。本質的に、および以下に詳細に論じられるように、このシステムは、ブラウザ７において実行されるAdControllerエージェントから発生する要求に応答して、所与の広告を選択し、次いで、そのエージェントにより制御される「礼儀正しい」態様にて、その広告を形成するメディアファイルおよびプレイヤーファイルをそのエージェントにダウンロードしてブラウザディスクキャッシュに記憶させる。Javaアプレットは、現在は、Javaプログラミング言語自身に固有の制約の下、ファイルの検索をそのアプレットそのものを供給した同一のインターネットホストから行なうことに制限されているので、広告に対するシステム２５への要求ならびにそのシステム２５により供給される結果のメディアファイルおよびプレイヤーファイルは代理サーバとしてのエージェントサーバ１５を介して経路制御される。

【００６０】

有利なことに、我々の発明的な技術は広告コンテンツをウェブコンテンツページ（以下、「参照」ページとも称される）から完全に「結合を解除」する。これにより、次いで、我々の技術は、どのような広告コンテンツも参照ウェブページに含む必要なくメディアリッチな広告を表示することができる。この「結合の解除」はＨＴＭＬタグがコンテンツウェブページに含まれることにより実現されるが、それは、後者がブラウザによりダウンロードされ解釈され実行されると、実際にエージェントをロードしてそのインスタンスを生成し、次いで、広告ファイルを、そのタグにより特定される広告管理システムから検索する。したがって、広告ファイル（メディアファイルおよびプレイヤーファイルの両方）はそれらの参照ウェブページから完全に独立して維持され得、しかも、有利なことに、前者に対してなされるどのような変更も、後者に含まれるＨＴＭＬ符号に影響しない。

【００６１】

特に、ＨＴＭＬタグ４０（必要であれば、このタグを他のＨＴＭＬタグと区別するために、それは以下「広告タグ」とも称される）は、コンテンツプロバイダによって、各参照ウェブページたとえばここではページ３５を構成するＨＴＭＬコードに埋め込まれる。一般に、このタグがこのページに対する既存のＨＴＭＬコード（図２Ａおよび図２ＢにおいてはＨＴＭＬコード部35_Aおよび35_Bとして示される）に対しどのように位置決めされるかは重要ではない。有利なことに、このタグに対応すべくこれらコード部に何らかの変更を行なう必要はたとえあったとしても非常に稀である。下の表１に示され再生されるよう

に、このタグは、典型的にはウェブページにおいて１行を占め、スクリプトを実施する。

【００６２】

【表１】

```
<SCRIPT SRC=http://unicast.com/loadad.js>
AdServer="http://AdManagement system"
</SCRIPT>
```

表１――広告タグ

【００６３】

広告タグの一部（「SRC=http://unicast.com/loaded.js」）は、ブラウザによって実行されると、JavaScriptファイル（「loaded.js」と名づけられる）をエージェントサーバからダウンロードする。このファイルは、次いで、スクリプトとしてブラウザにより解釈され実行される。図２Ａおよび図２Ｂにおいて示されるブロック２００によりシンボル化して示されるように、このスクリプトの実行の効果は、そのスクリプトにより動的に書込まれるアプレットタグを参照ウェブページに広告タグ４０の代わりに置換することにより、ブラウザディスクキャッシュにある、ここでは参照コンテンツページ３５′である修正されたウェブページを形成することである。万一ユーザが次いで自分のブラウザでページ３５′に対するＨＴＭＬソースコードを表示する場合には、そのスクリプトは、動的書込に関連づけられる特徴を呼出すことにより、これらタグを見えないように完全に隠す。これにより、次いで、ユーザは、ある程度、エージェントのソースおよび広告管理システムを容易に確認することを妨げられる。まとまって、これらアプレットタグはTransition Sensorアプレット２１０を形成する。このスクリプトは、以下に詳細に記され以下において表２に再生されるように、ブラウザに常駐するJava仮想マシン（Javaインタープリタ）によって解釈され実行されると、持続的にTransition Sensor自体をロードし次いでそれのインスタンスを生成し、次いで、Transition Sensorがそのエージェントの残りをクライアントブラウザにおいてロードしそのインスタンスを生成する。

【００６４】

【表２】

```
<applet code="com.unicast.adcontroller.tools.TransitionSensor"
codebase="http://www.unicast.com/java/classes/"
align="baseline" width="0" height="0" name="TransitionSensor"
archive="adcontroller.jar">
<param name="adURL"
value="http://www.unicast.com/media/fireworks_01_ad_descriptor.txt">
<param name="cabbase" value="adcontroller.cab">
</applet>
```

表２――TRANSITION SENSORアプレット

【００６５】

Transition Sensorアプレット内の属性ＣＯＤＥの値は、このアプレットを表示するときにクライアントブラウザによってTransition Sensorを起動するために実行されるJava executable［JAVA実行可能ファイル］を特定する。このexecutableは、適切なJavaクラスを介して実装されるが、元々はその関連のJavaソースコードファイルからコンパイルされた。「＜WIDTH＞」および「＜HEIGHT＞」とラベル付けされたタグは併せてブラウザ７により表示されるようにウェブページの矩形部分を特定し、そこにアプレットが表示される。

ここではその部分は存在していないので何も表示されない。このようなアプレットは目に見えない形でインターネットを介してクライアントＰＣに配信され得、ユーザにより補助されるインストールを全く必要としない。

【００６６】

この広告タグの他の部分（“AdServer=”http://AdManagement__system”）はある特定の広告システム、ここでは例示的にシステム２５として示されるシステムのＵＲＬ（「AdManagement__system」はその特定のシステムのウェブアドレス（ＵＲＬ）を表現する）を参照し、そこからエージェントによって広告がダウンロードされることになる。以下でわかるように、Transition Sensorアプレットはその実行中においてこのＵＲＬを広告ダウンロード要求の一部としてAdControllerエージェントの残りに送ることにより、後で、適切な広告ファイルが、これも以下で記載されるように、広告を隙間でプレイするのに必要なシステムからダウンロードされる。

【００６７】

広告のプレイが、アプレットタグの動的書込をサポートしないクライアントブラウザ（具体的にはマイクロソフト・インターネット・エクスプローラのバージョン３）上において行なわれる場合、アプレット２１０はコンテンツプロバイダによって各参照ウェブページ内に広告タグ４０の代わりに挿入される必要があるだろう。残念なことに、Transition Sensorアプレット２１０は、エージェントサーバと、実際の広告との双方を、そのソース要素（このアプレットにおいて特定される「AdDescriptor」ファイルのコンテンツを介する――これについては後に詳細に説明する――）のＵＲＬによって識別する。ブラウザ技術は急速に進展しつづけ、それに伴なって大半のユーザが継続的に自分たちのブラウザをアップブレードし続けるため、現在使用中の大半のブラウザ、および近い将来にはほとんどすべてのそのようなブラウザはそのような動的書込をサポートするであろう。したがって、我々は、アプレット２１０をどのような参照ウェブページに対しても埋め込むことはほとんどおよびまもなくすれば本質的に全く必要ないとみており、したがって、エージェントサーバおよび広告ソース情報の開示を制限しながら挿入コスト、労力および時間を最小限に抑える。

【００６８】

エージェントは、その実行中に、所与の広告に対する、広告管理システム２５から発生される広告ファイル（メディアファイル、および必要な場合にはプレイヤーファイル）をブラウザディスクキャッシュに（メディアファイルもブラウザＲＡＭキャッシュに書込まれる状態で）「礼儀正しく」かつ目に見えない態様でダウンロードし、次いでそれらメディアファイルをブラウザを介して隙間ごとにかつユーザのクリックストリームに応答してプレイする。

【００６９】

有利なことに、エージェントは、クライアントブラウザにおいて、任意の参照ウェブページ内のコンテンツからは独立して動作する。一旦ロードされスタートされると、エージェントは、並列に、標準的なブラウザ機能で実行され、継続的かつ目に見えない態様で広告を要求しそれをクライアントコンピュータ（たとえばパソコン――ＰＣ）にあるローカルハードディスク（「ブラウザディスクキャッシュ」）にあるブラウザディスクキャッシュおよびメディアファイルの場合にはブラウザＲＡＭキャッシュの中にダウンロードし、隙間でそれら広告をプレイする。

【００７０】

次に、上記を考慮しつつ、具体的には図１Ｂおよび１Ｃを参照し、本発明の使用に関連する基本的なコンピュータ間動作およびクライアントＰＣで生じる基本的な付随する処理ステップについて説明する。

【００７１】

ユーザはまず、ブラウジングセッションを開始するために、クライアントブラウザ７を呼出す。ブラウザは、実行を開始し、最初のウェブページとして（このページの選択を参照番号３１で示す）、ユーザが以前に特定しそのＵＲＬがブラウザに格納されている、過去

のいわゆる「デフォルト」コンテンツページの、または、ユーザが手で入力したコンテンツページのアドレスを受ける。次に、クライアントブラウザは、ブロック３３で示すように、このページに対するファイルをフェッチせよという要求を発行する。この要求はそのページのＵＲＬ（すなわちそのファイルネームを含む完全なウェブアドレス）を含む。簡潔にするために、このページに対するファイルはウェブサーバ１３にあると仮定する。また、ページ３５が要求されると本発明に従う関連の隙間広告が呼出されると仮定する。ライン３４で示すサーバ１３への要求に応答し、この特定のサーバは、ライン３６で示すようにページ３５に対するファイルをクライアントＰＣ５にダウンロードする。ここでこのファイルに格納されたコーディングは広告タグ４０を含む。このタグのコンテンツの例を、破線のブロック４５ならびに図２Ａおよび２Ｂに示す。

【００７２】

図１Ｂおよび１Ｃに示すように、このファイルが受取られると、ブラウザ７は、ブロック５２で示すように、タグ４０を含むページ３５のＨＴＭＬコードを解釈して実行し、エージェントダウンロード／インスタンス生成／実行動作５０に示した動作を行なう。最終的にはこれらの動作の結果、クライアントブラウザで、AdControllerエージェントのダウンロード、インスタンス生成および開始が行なわれる。一般的に、ブラウザは、広告タグの実行に応答して、ライン５４で示すように、エージェントサーバ１５に対しAdControllerエージェントをダウンロードせよという要求を発行する。図１Ｅおよび１Ｆでさらに詳細に示し以下で簡単に説明する種々のプロセス間動作を通して、サーバ１５は、ライン５６で示すように、必要なファイルにアクセスしこれをダウンロードし、AdControllerエージェントをインストールし、これがクライアントＰＣのブラウザ７下で実行される。このエージェントに対するファイルがクライアントＰＣのブラウザのディスクキャッシュにダウンロードされると、ブラウザにより、ブロック５８で示すように、エージェントのインスタンス生成および実行開始がなされる。エージェントが実行を開始すると、動作５０は実際上終了する。

【００７３】

次に、図１Ｅおよび１Ｆでさらに詳細に示す動作５０を参照して、ブラウザ７は、これらの動作に入ると、ブロック１１０で示すように広告タグ４０を実行する。ブラウザは次に、ライン１１５で示すように、エージェントサーバ１５に対し、特定したJavaScriptファイル（たとえば「loadad.js」と名付けられる）をダウンロードせよという要求を発行する。このファイルは、広告タグの第１部分として特定されている。この要求に応答して、サーバ１５は、ライン１２０で示すように、この特定のファイルをブラウザ７にダウンロードし、ここでこのファイルは適切にキャッシュされる。このファイルが完全にダウンロードされると、Java仮想マシン（ブラウザに一体化されブラウザと互換性がありかつブラウザが実行可能なコードを発生するJavaインタープリタ）は、このファイルを解釈および実行する。ブロック１２５で示すように、ブラウザは次に、このスクリプトについて解釈されたコードを実行し、今度は、アプレットタグが、一般的には図２Ａおよび２Ｂに示し先に説明した態様で、広告タグの代わりにウェブページ３５に書込まれる。これらのタグは全体としてTransition Sensorアプレット２１０を形成し、広告タグ４０の第２部分で特定されるように特定の広告管理システムへの参照を含む。

【００７４】

これらのタグが動的にコンテンツウェブページ３５に書込まれると（図２Ａおよび２Ｂに示す変更バージョン３５′が生まれる）、Transition Sensorアプレット２１０のインスタンス生成が行なわれ、次にアプレット２１０が実行される。特に、ブラウザ７は、Transition Senorアプレットについて実行可能なコードが、以前にブラウザのディスクキャッシュにダウンロードされているかどうかを判断する。このコードがダウンロードされていない場合、または、このコードの更新バージョンがエージェントサーバ１５にある場合、ブラウザは、ライン１３０で示すように、エージェントサーバから実行可能なTransition Sensorコードの最新バージョンをダウンロードせよという要求を発行する。サーバ１５は、この要求に応答して、ライン１３５で示すように、Transition Sensorコードの最新

バージョンに対するファイルをブラウザにダウンロードし、今度は、そのファイルがブラウザのディスクキャッシュに格納される。ブラウザはその後、ブロック１４０で示すように、Transition Sensorアプレットのインスタンス生成を行ない、このアプレットの実行を開始する。この後者のアプレットは、その最初の実行の一部として、アプレットレジストリのインスタンス生成を行なう。このレジストリは、エージェント内で、構成要素であるTransition SensorおよびAdControllerアプレット間のアプレット間通信のためのメカニズムを提供する。

【００７５】

その後、Transition Sensorアプレットは、これもまたブロック１４０で示すように、AdControllerアプレットをブラウザを通してブラウザのディスクキャッシュからロードしようとする。ブラウザはこのために、初めに、AdControllerアプレットがブラウザのディスクキャッシュにダウンロードされているかどうか、または、この特定のアプレットの更新バージョンがエージェントサーバ１５にあるかどうかを判断する。このアプレットの更新バージョンが、ブラウザのディスクキャッシュに以前にダウンロードされたものと関連してエージェントサーバにある場合、または、AdControllerアプレットがこのキャッシュに全くダウンロードされていない場合、ブラウザは、ライン１５０で示すように、エージェントサーバ１５からAdControllerアプレットの最新バージョンをダウンロードせよという要求を発行する。サーバ１５はこの要求に応答して、ライン１５５で示すように、AdControllerアプレットの最新バージョンに対するファイルをクライアントブラウザにダウンロードし、今度は、このファイルがブラウザのディスクキャッシュに格納される。最後に、ブロック１６０で示すように、Transition Sensorアプレットは、AdControllerアプレットのインスタンス生成を行ないこのアプレットを開始し、その後、自身のためにかつAdControllerアプレットのためにアプレットレジストリに適切なエントリを設ける。

【００７６】

図１Ｂおよび１Ｃを再び参照して、動作５０が完了しエージェントがブラウザ７の下で実行していると、AdControllerアプレットは、ブロック６０で示すように、エージェントサーバ１５を介して、例として広告管理システム２５といった広告管理システムから、広告タグ４０に特定されたAdDescriptorファイルをダウンロードせよという要求を発行する。この要求は、広告タグ４０に含まれる広告管理システムのＵＲＬを含む。現在、Javaアプレットは、Javaプログラミング言語そのものに固有の制約によって制限され、アプレットそのものを提供するのと同じインターネットホストからしかファイルを取寄せることができない。したがって、この要求を、広告管理システム２５がある広告サーバ２０に向けるのではなく、ライン６２で示すように、クライアントＰＣ５と広告サーバ２０との間でプロキシサーバの役割を果たすエージェントサーバ１５に向ける。この要求および結果として得られた広告（メディアおよびプレイヤーを含む）ファイルは、エージェントサーバ１５を通してクライアントＰＣに提供される。したがって、エージェントサーバは、この要求を受けると、ライン６４で示すように、広告サーバ２０に送る。

【００７７】

このAdDescriptorファイルの要求に応答して、広告管理システム２５は、クライアントＰＣ５に配信すべき特定の広告を選択する。この選択は、予め規定されたものに基づいてもしくはランダムに、または、ユーザの嗜好に基づいてもしくはそのときブラウザ７を操作しているユーザから以前に収集しそのユーザと関連付けた、ユーザに特有の他の情報に基づいて、選択できる。このような、過去の購買パターンといったユーザ特有の情報を、クライアントＰＣで予め適切に収集し、先に広告管理システム２５にアップロードしここで処理して、システム２５がAdDescriptor要求を受けたときに、そのときにクライアントＰＣ側にいるユーザを特定的に対象とする適切な広告を選択およびダウンロードできる。どのような場合でも、システム２５が一旦何らかの選択基準を採用しこの基準によって広告を選択すると、ライン６６で示すように、対応するAdDescriptorファイルがエージェントサーバ１５（ここではプロキシサーバ）にダウンロードされ、今度はライン６８で示すように、エージェントサーバがそのファイルをこのときウェブブラウザ７で実行しているAd

Controllerエージェントに与える。

【００７８】

少し話は反れるが、この選択された広告について、AdDescriptorファイルは目録を含むテキストファイルである。すなわち、ファイル名およびこうしたファイルが存在する対応のネットワークロケーション（ＵＲＬ）、ならびに、ウェブブラウザ７を通して広告全体をユーザに対しプレイするのに必要なプレイヤー命令および構成パラメータのリストである。図２０は、PointCast Java広告についての典型的なAdDescriptorファイル２００のコンテンツを示す。具体的にはファイル２０００のSection４Ｃに示すように、このAdDescriptorファイルには、ファイル名のリストがあり、ここにはこの広告のコンテンツを構成するすべてのメディアファイルの広告管理システム上の部分的なアドレスが添えられ、このファイルのSection１には、すべてのメディアファイルをプレイするのに必要なすべてのJavaプレイヤーファイルがある。このファイルはまた、それぞれここではSection３および４Ｂで示すように、種々のメディアファイルをプレイする順序、および、各プレイヤーの動作を適切に構成して対応する各メディアファイルをプレイするのに必要な種々の構成パラメータを特定する。

【００７９】

AdDescriptorファイルは、メディアおよびプレイヤーファイルをここではページ３５である参照しているウェブページから完全に切離すことによって、データ抽象化を実現し、確実に、広告のコンテンツそのものがそのプレゼンテーションを呼出したコンテンツウェブページと全く無関係になるようにする。この抽象化によって、当技術は、特に複雑なメディアファイルおよびプレイヤーのセットを必要とするリッチなウェブ広告を配信するための、非常に効果的で汎用化されかつ非常に柔軟なメカニズムを提供することができる。この抽象化を用いることで、本発明の技術は、現在および未来のメディアフォーマットを、その要求にかかわらず、配信メカニズムとしてJavaアプレットに依存する各社固有のストリーミングおよび他のコンテンツ配信技術を含めて、すべてユーザにはわからないように扱うことができる。さらに、AdDescriptorファイルは、そのとき使用している異なるブラウザ、クライアントオペレーティングシステム、または計算プラットホーム（これらのなかでメディアおよび／またはプレイヤーファイルの異なるバージョンを要求するものがあるという程度で）に対するメディアおよびプレイヤーファイルを詳しく示した別のリスト（図２０に示したファイル２０００には含まれていない）を含むことができる。したがって、当技術は容易に、異なる種々のクライアントコンピュータおよびブラウジングプラットホームとともに機能できる。

【００８０】

AdDescriptorファイルがエージェントサーバ１５を介してクライアントＰＣにダウンロードされると、AdControllerは、図１Ｂおよび１Ｃに示したブロック７０で示すように、AdDescriptorファイルに特定されている、各メディアおよびプレイヤーファイルを、このファイルがクライアントＰＣのハードディスクになければ、ブラウザのディスクキャッシュに「ポライトに（礼儀正しく）」ダウンロードする。いわゆる「礼儀正しい」ダウンロードを通して、メディアおよびプレイヤーファイルは、ブラウザのアイドルタイム期間中にブラウザ７にダウンロードされ、このダウンロードは、ユーザがブラウザ７に新たなコンテンツウェブページへのナビゲートを命令した後は、続く各隙間期間中はサスペンドされる。この態様で、完全にダウンロードされた広告がブラウザキャッシュから隙間でプレイされている間に、通信リンク９の全帯域にわたって新たなコンテンツページがダウンロードされる。通信リンクは好都合なことにただウェブページコンテンツを送るだけで各隙間期間中は空いているので、コンテンツページのダウンロードを迅速に行なうことができる。隙間期間の発生のために、AdControllerアプレットが広告ファイルのダウンロードをサスペンドすると、この期間が終了した後に、このアプレットは、ダウンロードが停止した、このファイル中のロケーションでダウンロードを再開し、これによって通信帯域幅を保ちダウンロード時間を短縮する。

【００８１】

特に、ブロック７０で示す動作の一部として、AdControllerアプレットは、AdDescriptorにリストされたファイルのうちどのファイルがそのときにクライアントＰＣ５のハードディスクにないのかを判断する。この判断がなされると、このアプレットは、ライン７２で示すように、エージェントサーバ１５に対しこれらのファイルのうち第１のファイルをフェッチせよという要求を発行する。ここでもプロキシサーバの機能を果たしているエージェントサーバは、ライン７４で示すように、そのファイルがある、インターネット１０のどこかの場所にあるネットワーク化されたサーバからこのファイルをフェッチせよという要求を発行する。簡潔にするために、ここではこういったファイルすべてがサーバ２０にあり、広告管理システム２５を通してアクセス可能なものであるとする。したがって、システム２５はサーバ２０を介して、ライン７６で示すようにこの最初の広告ファイルを含む応答をエージェントサーバ１５に対して発行する。今度はこのエージェントサーバが、ライン７８で示すように、この特定のファイルをクライアントブラウザ７にダウンロードしてブラウザのディスクキャッシュに格納できるようにする。広告ファイルのダウンロードは、この態様で、ライン８８で示すように、広告について最後に要求されたファイルがエージェントサーバ１５を介してクライアントＰＣ５のブラウザのディスクキャッシュにダウンロードされるまで続けられる。

【００８２】

共通の広告に対する広告ファイルがダウンロードされているとき、Transition Sensorアプレットもまた、ブロック９０で示すように、現在のユーザが発生するクリックストリームをモニタし、ユーザが開始したページ移行を検出する。このような移行は通常ユーザがマウスをクリックすることにより生じ、こうして隙間期間が始まると、AdControllerは、これもブロック９０で示すように、完全にキャッシュされた広告（そのすべてのメディアおよびプレイヤーファイルが既にダウンロードされていると仮定する）を、関連のAdDescriptorファイルで特定されている態様でかつここで特定されたプレイヤーを用いて、プレイする。また、この隙間期間の開始時に、ブラウザは、ブロック９０で示すように、ユーザが移行したいと望んでいる次に続くウェブページをフェッチせよという要求を発行する。広告が完全にプレイされた後、次に続くコンテンツウェブページが完全にダウンロードおよびアセンブルされるまで、およびユーザが広告ウィンドウを閉じるまで、のうち、いずれか最初に発生したときに（AdDescriptorファイルが広告を途中で終了できると特定していると仮定する）、コントロールが経路９４で示すようにクライアントブラウザに戻され、広告タグが中にある次のコンテンツページを形成するＨＴＭＬコードのダウンロードおよび解釈ならびにそれに続く実行の完了を待ち、そのページに対するエージェントのダウンロード／インスタンス生成／実行動作５０の呼出を行ない、以降同様に続く。

【００８３】

Transition SensorおよびAdControllerアプレットは、各々適切なJavaクラスを通して実現され、サイト内の異なるコンテンツページ、異なるウェブサイト、および連続するブラウザセッションを通して、ブラウザのディスクキャッシュへの格納を通し、全体として存続する。これらのアプレットのうちいずれかが動作５０を通して完全にダウンロードされた後、そのアプレットが、ユーザがウェブ上のウェブサイトをナビゲートしている際にブラウザのディスクキャッシュからフラッシュされていない場合、そのアプレットに対するファイルは、エージェントサーバ１５からダウンロードされるのではなくそのキャッシュからロードされ、次回は、そのアプレットは、たとえば次にユーザが現在のブラウザセッション中にまたは後続のセッション中にナビゲートしているときに、広告タグ４０を含む連続するいずれかのコンテンツページにとって必要になる。

【００８４】

クライアントブラウザ７は、広告タグ４０を含む後続のコンテンツページに出会うと、最初にかつ自動的にエージェントサーバ１５に問合せ、Transition Sensorアプレットについて実行可能なコードが、もし以前にブラウザのディスクキャッシュにダウンロードされているならば、更新バージョンによって上書きされていないことを確かめる。そのときこのような更新バージョンがある場合、ブラウザは、エージェントサーバから更新ファイル

をまとめてダウンロードし、ブラウザディスクキャッシュにある各Transition Sensorアプレットファイルを必要な程度更新バージョンと置き換える。その代わりとして、Transition Sensorアプレットがブラウザのディスクファイルにまだダウンロードされていない場合、ブラウザは、エージェントサーバから、Transition Sensorアプレットに必要なファイルすべてを、そのキャッシュにダウンロードする。Transition Sensorアプレットは実行を開始すると、ブラウザを通してAdControllerアプレットをロードする。ブラウザはこのために、必要であれば、Transition Sensorについて行なったのと同じ態様でエージェントサーバから更新バージョンを得る。その結果、エージェント（具体的にはTransition Sensorおよび／またはAdControllerアプレット）が最後にクライアントブラウザにダウンロードされて以来、そのエージェントに対してなされた修正または向上は、自動的にかつユーザ側からはわからない状態で、ブラウザがある広告タグを含むウェブページに次に出会ったときに、ブラウザに送られかつそのディスクキャッシュにダウンロードされる。この態様で動作を行なうことによって、ユーザは、完全にかつ好都合なことに、最初にアプリケーションプログラムをロードおよびインストールして広告を得ること、および／または後にそのプログラムを更新する必要から解放される。

【００８５】

具体的に、Transition Sensorエージェントのページ間永続性は、Java「シングルトン」デザインを用いて可能である。シングルトンデザインは、単一のオブジェクトのみの作成を可能にし、Javaクラスを静的なものであると宣言することによって実現される。すべてのアプレットはJava仮想マシン内の同じインスタンスで実行しているため、すべてのアプレットおよびそれらに関連するコードはすべての静的クラス変数を共有する。静的アプレットレジストリクラスは、Transition Sensorアプレットによりその実行時に自動的にそのインスタンスが生成され、アプレットレジストリを実現することによって、Transition SensorおよびAdControllerアプレットおよびこれらのスレッド間のアプレット間通信すべてがもたらされる。アプレットレジストリクラスは、「loadAdController」メソッドを実現し、これが永続するAdControllerアプレットのインスタンスを生成する。このメソッドを通して、Transition Sensorアプレットは、AdControllerアプレットを、後者のアプレットがこのアプレットのそのバージョンに関して更新されてブラウザのディスクキャッシュにある場合、またはブラウザのディスクキャッシュにない場合にのみ、ダウンロードする。AdControllerアプレットは次に、見えない広告ダウンロードおよびプレイメカニズムを全体として実現する自身のスレッドのインスタンスを生成する。

【００８６】

AdControllerアプレットそのものは、アプレットレジストリシングルトンオブジェクトにより作成され、実行時エージェント実行モジュールを集合的に構成する他のすべてのオブジェクトを生成する。このアプレットは、標準アプレットクラス定義を、従来はクライアントブラウザにおいて実現されていたライフサイクルの標準Javaアプレットinit（初期化）、start、run、stopおよびdestroyメソッドのオーバライドを、対応する代替メソッドにより行ない、拡張する。この代替のstop（ストップ）メソッドにより、AdControllerアプレットに対する実行停止というブラウザの従来の応答が、ブラウザがこのstopメソッドを呼出してこのアプレットのライフサイクルを終了させようとするときは常に生じないことが確実になる。したがって、有利にも、連続するコンテンツページにわたるエージェントの持続性が得られる。結果として、エージェントは、ユーザがブラウザそのものの実行を終了させる（ブラウザを閉じる）まで、実行を続ける。

【００８７】

このように、エージェントはバックグラウンドでわからないように持続かつ機能しており、共通のウェブサイトのページにわたるおよびウェブサイトにわたるユーザのナビゲートからは独立し、かつこれにはわからない。この点で、エージェントは効果的に、クライアントブラウザが実現する通常のＨＴＭＬおよびＨＴＴＰ動作と並列にかつこれにわからないようにバックグラウンドプロセスを実現する。

【００８８】

説明および添付の図面を大幅に簡潔化するために、ここでの記載から意図的に、AdControllerエージェントを構成する特定のJavaクラスを省略しており、かつ、広告を再生のために待ち行列に入れる速度を高めるための、マルチスレッド化されたパイプライン化に基づきこれらのクラスを処理するための付随するソフトウェアアーキテクチャも省略している。このような詳細は本質的に従来通りなので、当業者にとっては、これを本発明の実現のために使用することは、容易に明らかになるであろう。

【００８９】

Ｂ．クライアントＰＣ

図３は、クライアントＰＣ５のブロック図を示す。

【００９０】

図示のように、クライアントＰＣは、入力インターフェイス（Ｉ／Ｆ）３２０、プロセッサ３４０、通信インターフェイス３５０、メモリ３３０、および出力インターフェイス３６０を含み、すべて従来通りバス３７０によって接続される。メモリ３３０は一般には異なるモーダリティを含み、例として一時データおよび命令記憶のためのランダムアクセスメモリ（ＲＡＭ）３３２、ユーザのコマンドに応じてフロッピーディスケットと情報交換するためのディスケットドライブ３３４、および典型的には本質的に磁性であるハードディスクを通して実現される不揮発性大容量記憶３３５を含む。大容量記憶３３５は、ＣＤ－ＲＯＭ、または適切な光記憶媒体から情報を読取る（かつ情報を適切な光記憶媒体に書込む）ための他の光媒体リーダ（具体的には図示せず）（またはライタ）を含み得る。大容量記憶は、オペレーティングシステム（Ｏ／Ｓ）３３７およびアプリケーションプログラム４００を格納し、後者は例として本発明の技術を実現するブラウザ７（たとえば図１Ｂおよび１Ｃ参照）を含む。図３に示すＯ／Ｓ３３７は、WINDOWS NT、WINDOWS 95、またはWINDOWS 98オペレーティングシステム（「WINDOWS NT」、「WINDOWS 95」および「WINDOWS 98」はワシントン州RedmondのMicrosoft Corporationの商標である。）といった、従来のオペレーティングシステムにより実現し得る。このことから、Ｏ／Ｓ３３７の構成要素についてはすべて無関係であるため説明を行なわない。アプリケーションプログラム４００の１つであるブラウザがＯ／Ｓの制御下で実行するものであることを述べておくだけで十分であろう。

【００９１】

入来する情報は、例として２つの外部ソースから生じ得る。すなわち、たとえば通信リンク９を通してインターネットおよび／または他のネットワーク化された設備から通信インターフェイス３５０にネットワークで与えられる情報、または経路３１０を介し専用入力ソースから入力インターフェイス３２０に与えられる情報である。

【００９２】

専用入力は、たとえば外部データベースといった広範囲にわたるソースから生じ得る。加えて、ファイルまたはその中の特定のコンテンツの形態である入力情報もまた、情報を含むディスケットをディスケットドライブ３３４に挿入することによって得られ、このドライブから、クライアントＰＣ５はユーザから命令されるとそのディスケットからの情報にアクセスして情報を読出す。入力インターフェイス３２０は、入力情報の異なる専用ソース各々をクライアントＰＣ５に物理的に接続することおよびこれとのインターフェイスに必要な対応する電気接続をもたらす適切な回路を含む。オペレーティングシステムの制御下で、アプリケーションプログラム４００は、ネットワーク接続９または経路３１０を介し、コマンドおよびデータを外部ソースとやり取りし、プログラム実行中に典型的にはユーザが要求する情報の送受信を行なう。

【００９３】

入力インターフェイス３２０もまた、キーボードおよびマウスといったユーザ入力装置３９５をクライアントＰＣ５に電気的に接続しかつこれとのインターフェイスを行なう。従来のカラーモニタといったディスプレイ３８０および従来のレーザプリンタといったプリンタ３８５は、リード線３６３および３６７それぞれを介して出力インターフェイス３６０に接続される。出力インターフェイスは、ディスプレイおよびプリンタをコンピュータ

システムに電気的に接続しかつこれとのインターフェイスを行なうのに必要な回路を提供する。

【００９４】

さらに、クライアントＰＣ５の特定のハードウェア構成要素およびメモリ３３５内に記憶されるソフトウェアのすべて局面は、本発明を実現するモジュールは別にして従来通りでかつ周知であるため、ここではこれ以上の詳細な説明は行なわない。一般的には、エージェントサーバ１５および第三者広告サーバ２０は各々、クライアントＰＣ５と非常によく似たアーキテクチャを有する。

【００９５】

Ｃ．ソフトウェア

１．アプリケーションプログラム４００

図４は、クライアントＰＣ内に常駐するアプリケーションプログラム４００の高レベルブロック図を簡単にしたものを示す。

【００９６】

図示のように、アプリケーションプログラムは、関連する限りにおいては、ブラウザ７および常駐Javaプレイヤーファイル４１０を含む。すなわち、クライアントＰＣのハードディスクに既にインストールされているJavaメディアプレイヤーのためのファイルを含む。このプレイヤーは例としてオーディオ、ストリーミングオーディオ、ビデオおよびマルチメディアプレイヤーを含み得る。

【００９７】

ブラウザ７は、AdControllerエージェント４２０を含み、これが実行のためにブラウザのキャッシュに完全にロードされると、ブラウザディスクキャッシュ４３０およびJava仮想マシン４４０（関連のある限りでは先に説明したもの）を含む。上記のように、このエージェントは、ユーザによりブラウザ７が異なるウェブコンテンツページまたは異なるウェブサイトを通して移行するときは常に持続しており、こうしたページおよびサイトから独立してかつこれらにわからないように機能する。AdControllerエージェントは、エージェント内のアプレット間通信を容易にするためのアプレットレジストリ４２６を含む。

【００９８】

AdControllerエージェントは２つのアプレットすなわちTransition Sensorアプレット４２２およびAdControllerアプレット４２４を含む。上記のように、Transition Sensorアプレットは基本的な３つの機能を果たす。第１に、このアプレットは、AdControllerアプレットのロード、インスタンス生成および開始を行なう。第２に、Transition Sensorアプレットは、ここではサーバ２０である広告サーバのインターネットアドレスを送り、広告を、具体的にはダウンロードされ後に提示される、その広告についてのAdDescriptorファイルを、要求する。最後に、Transition Sensorアプレットは、関連のクリックストリームモニタ（このアプレットが実現するTransition Sensorが行なう。）を通して、クライアントブラウザ７側のユーザが次に続くウェブページのダウンロードおよび表示を要求するためのたとえばマウスクリックによる確認動作をいつ行なったのかを判断し、このイベントをAdControllerエージェントに通知する。このイベントは、続く隙間期間の開始を知らせる。

【００９９】

AdControllerアプレット４２４はコンテンツページには埋込まれていないものであり、ブラウザ７下で実行するがブラウザ７によって制御されるものではない。このアプレットも上記のように、いくつかの基本機能を果たす。第１に、このアプレットはそのエージェントに対する実行時エージェント実行モジュールを集合的に形成するその他すべてのオブジェクトを生成する。上記のように、これは、ライフサイクルの標準Javaアプレットinit、start、run、stopおよびdestroyメソッドのオーバライドにより標準Javaアプレットクラス定義を拡張することを含む。第２に、AdControllerアプレットは、クライアントコンピュータで実行しているクライアントブラウザを通して、広告（メディアを含み必要な場合はプレイヤーを含む。）ファイルをブラウザディスクキャッシュに、ブラウザ側のユーザ

にはわからない態様で「礼儀正しく」ダウンロードする。最後に、AdControllerアプレットは、異なるウェブページを通る通常のユーザナビゲーションと関連するユーザのクリックストリームに応答して、クライアントブラウザを通して隙間で広告をプレイする。

【０１００】

ブラウザディスクキャッシュ４３０は、ダウンロードされたAdDescriptorファイル４３３および添付のダウンロードされたメディアおよびプレイヤーファイル４３７を格納する。

【０１０１】

２．AdControllerエージェント４２０

図５は、AdControllerエージェント４２０の高レベルブロック図を示す。

【０１０２】

図示のように、このエージェントは具体的に、Transition Sensorアプレット４２２、AdControllerアプレット４２４およびアプレットレジストリ４２６を含む。

【０１０３】

先に一般的に説明したように、Transition Sensorアプレットは、その機能の１つとして、Transition Sensorを実現し、これによって、ユーザのナビゲーションのクリックストリームのモニタを通し、ユーザが開始した新たなウェブページへの移行を検出し、これに応答して、対応するTransition Sensorイベントを生成する。このような移行は、同じサイトまたは別のサイトで新たなコンテンツページに移動するために、現在表示されているコンテンツページ上に現われるホットリンクを活性化させるためにユーザが実際にマウスクリックを開始することまたはキーを押すことに応答して発生する。別のこのような移行は、ユーザが訪れたばかりのウェブページの格納されたヒストリが状態を変えたときには常に生じる。後者は、JavaScript機能が感知するもので、これは、訪問を受けたウェブページのＵＲＬの、ブラウザディスクキャッシュ４３０に格納されたヒストリをモニタし、ヒストリの状態に変化が起きたときは常にイベントを生じさせる。簡単に参照できるようにするために、「クリックストリーム」という用語をここではまとめて、マウスクリック、キーを押すことまたはヒストリ状態の変化といった、ユーザが開始する新たなコンテンツページへの移行を含むものと定義する。

【０１０４】

Transition Sensorイベントを用いて、広告のプレイを、この広告に対するメディアおよびプレイヤーファイルすべてが完全にこのときまでにブラウザのディスクキャッシュ４３０にキャッシュされている場合にのみ、トリガする。それ以外の場合は、その広告のプレイを、こういったファイルすべてがキャッシュされ広告の表示の準備ができるまで遅らせ、ここで重要なことは次にユーザが移行を開始するのに応答してプレイされる。

【０１０５】

クライアントブラウザ７は、ライン５０５および５１０それぞれで示すように、Transition Sensorのinit（初期化）およびstart、ならびにstopイベントを発生する。ブラウザはinitおよびstartイベントを生じさせ、Transition Sensorアプレットを初期化（すなわちロードおよびインスタンス生成）しこれを開始する。ブラウザはstopイベントも生じさせるが、これは、ユーザが開始したページ移行の検出に応答してTransition Sensorによる標準ブラウザstopメソッドと置換えられたTransition Sensor stopメソッドを通したものである。これらのイベントは、アプレット４２２の状態を制御する。Transition Sensorアプレット４２２は直接、AdControllerアプレット４２４とライン５３５で示すように通信し、広告サーバのインターネットアドレスを送り、ライン５３０で示すようにアプレットレジストリ４２６を通して間接的に通信する。レジストリ４２６はライン５４０で示すように情報をAdControllerアプレット４２４に送る。

【０１０６】

上記のように、AdControllerアプレット４２４は、標準Javaアプレットの、ライフサイクルのinit、start、run、stopおよびdestroyメソッドをオーバライドすることにより、標準Javaアプレットクラス定義を拡張する。このことによって特にstopメソッドの場合（これは図１８と関連付けて以下で説明する）、AdControllerアプレットを、ユーザが連続す

るページおよびウェブサイトをナビゲートしている際に、ブラウザのディスクキャッシュ４３０で持続させる。

【０１０７】

好都合にも、AdControllerアプレットは、広範囲にわたる環境内で、アプレットそのもののコーディングの変更なしで容易に機能できる。これは、エージェントサーバ１５から、アプレットファイルの一部として、外部構成ファイル（具体的には以下で説明する図６Ａおよび６Ｂに示したファイル６２０）をダウンロードすることにより行なわれる。構成ファイル内のパラメータ値を適切に変更することによって、アプレット４２４の振舞いを所望の環境に適したものに容易に変更でき、この際異なる環境各々に対しアプレットの異なるバージョンを利用する必要はなく、これはさもなければ異なるソフトウェアクラスを要求するもので修正および再コンパイルが伴うものである。

【０１０８】

AdControllerアプレット４２４の実行は、Transition Sensorアプレット４２２が標準initアプレットメソッドを呼出すことによって開始され、これは外部構成ファイルをダウンロードするもので、その次にはその構成パラメータの抽出および保存が行なわれる。こうしたパラメータは、ライン５１５で示すように、AdControllerアプレットに、その実行中に与えられた現在の実行環境下の振舞いを規定するために与えられる。

【０１０９】

上記のように、AdControllerアプレット４２４は、広告（メディアを含み必要な場合はプレイヤーを含む）ファイルを、後に隙間でプレイされるすべての広告に対し、ブラウザ７を通してブラウザのディスクキャッシュ４３０に「礼儀正しく」かつわからないようにダウンロードする。広告がダウンロードされるデータ経路は図５において一点鎖線で示されており、広告のプレイについてのデータ経路はこの図面において点線で示される。

【０１１０】

具体的に、広告をダウンロードしてプレイするために、アプレット４２４はAd Pipeline５４５（これについては以下において図１４と関連付けて詳細に説明する）を実現する。Pipeline５４５は種々のスレッド（プロセス）およびデータ構造を実現し、これらは、集合的に広告ファイルをブラウザのディスクキャッシュ４３０にロードし（メディアファイルについてはブラウザのＲＡＭキャッシュにもロードする）、次に完全にダウンロードされた広告を表示する。Pipelineは、Ad Producer、Ad LocationおよびAd Downloaderプロセス（それぞれ図１５、１６および１７に示すプロセス１５００、１６００および１７００であり以下で詳細に説明するものである）、ならびにダウンロード待ち行列１４３０およびプレイ待ち行列１４７０（双方は図１４に示されており以下で詳細に説明するものである）を実現する。

【０１１１】

本質的には、Transition Sensorアプレット４２２が図５に示すようにAdControllerアプレット４２４にAdDescriptorファイルのＵＲＬを与えると、Ad Pipeline５４５は、一点鎖線５２０で示すように、遠隔の広告管理システムからエージェントサーバ１５（プロキシサーバの役目を果たす）を介してAdDescriptorファイルをダウンロードする。上記のように、このファイルは、広告全体を十分にプレイするのに必要なメディアおよびプレイヤーファイルの目録を含む。このAdDescriptorファイルがAd Pipeline５４５にダウンロードされると、Pipeline５４５は、ライン５２５で示すように、目録に特定された各ファイルを、そのファイルが既にクライアントのハードディスクに存在しているのでない限りにおいて、「礼儀正しく」ダウンロードする。次に、Pipeline５４５は、（必要な限りにおいて）ダウンロードが完了すると、AdDescriptorファイルをプレイ待ち行列に書込み、そこに特定された各ダウンロードファイルをブラウザのディスクキャッシュ４３０に書込む。このようにして、後続のアクセスに対し待ち行列に入れられた広告が形成される。

【０１１２】

Transition Sensorのstopイベントにより通知される、隙間期間の開始時に、AdControllerアプレットは、メディアおよびプレイヤーファイルに関し完全に待ち行列に入れられて

いる広告を隙間でプレイする。特に、その期間の最初に、Ad Pipelineは、プレイ待ち行列の先頭にあるAdDescriptorを取出す。AdDescriptorファイルで特定されている、その広告が要求するメディアプレイヤー５６５は、そのファイルにおいて対応するメディアファイルとともに特定されている順序で開始される。プレイヤーが生成する、ライン５７０で示された、結果として得られる処理後のメディアストリームは、ブラウザ７を通してユーザに表示される。メディアプレイヤー５６５は、クライアントのハードディスクに永続的に存在してもよい、すなわちエージェント４２０がダウンロードする場合を除き永続的に存在し（したがって図４で示すように常駐プレイヤーファイル４１０によって実現されている）、または、後でアクセスおよび使用するためにブラウザのディスクキャッシュ４３０に（ブラウザのＲＡＭキャッシュにも）Pipeline５４５によってダウンロードされてもよい（したがって図４に示すファイル４３７に格納される。）。

【０１１３】

広告が完全にプレイすると、AdControllerアプレット４２４は、図５に示すように、その広告に対する「ユーザ到達」についての適切なログエントリを確立する。広告ファイルは、ブラウザのディスクキャッシュにおいて、そのキャッシュが一杯になるまで保持され、この時点でこれらのファイルはそのキャッシュに格納された他のコンテンツファイルと同様、先入れ先出し（すなわち時間的順序）に基づいて削除される。メディアプレイヤー５６５、ブラウザ７およびブラウザディスクキャッシュ４３０はすべて構成要素として破線で示され、AdControllerエージェントが使用している場合は、エージェントそのものの中で単独で構成要素をなしているものとしては見ることができない。

【０１１４】

図６Ａおよび６Ｂは、AdControllerエージェント４２０が実行する処理動作６００の高レベルフローチャートをまとめて示したもので、この図の紙面の正しい配置を図６に示す。この図面に、また図８、９Ａおよび９Ｂ、１２、ならびに１４－１９に示した動作は、パイプライン化に基づいて複数の広告を処理するマルチスレッドを用いたアプローチを通しておこなわれるが、これらの図面すべてを簡潔にするために、各図面に示したシーケンシャルな処理フローは、１つの共通の広告を処理するものとする。スレッドおよびクラスについての説明は、当業者が、こうしたシーケンシャルな処理の流れがマルチスレッドを用いたJavaクラス方法を通していかにして実現されるかを十分に理解するのに必要な程度に行なわれる。

【０１１５】

ブラウザ７からのTransition Sensorのinitイベントに応答して、図６Ａおよび６Ｂに示すプロセス６００に入ると、ブロック６１０が実行される。このブロックを通して、Transition Sensorアプレット４２２は、アプレットレジストリに、AdControllerアプレットをロードするよう命令する。これが生じると、ブロック６１５が実行され、外部AdController構成ファイル６２０が、エージェントサーバ１５から取出される。その後、判断ブロック６３０を通して、エージェント４２０は、ブラウザ７がTransition Sensorのstartイベントを発生するまで、ＮＯ経路６３１をたどることによって待機する。このようなイベントが生じると、実行は、この判断ブロックから発生するＹＥＳ経路６３３を介し、ブロック６３５に進む。この後者のブロックを通して、AdControllerアプレット４２４は、広告管理システム（たとえばシステム２５）のインターネットアドレスを得、ここからエージェントはAdDescriptorファイル６４５を取出す。次にアプレット４２４は、このアドレスをAd Pipeline５４５に送る。Ad Pipelineはブロック６４０で示すように、このアドレスから、特にプロキシサーバの役割をするエージェントサーバ１５を通して、AdDescriptorファイル６４５を取出す。このファイルが取出されると、エージェントは、ブロック６５０を実行して、広告管理システム２５から（広告サーバ２０にある）、すべてのメディアおよびプレイヤーファイル６５５を（各ファイルがクライアントのハードディスクにない限りにおいて）「礼儀正しく」ダウンロードし、ブロック６６０を通して、これらのファイルをブラウザのディスクキャッシュ４３０（メディアファイルの場合はブラウザのＲＡＭキャッシュ）に格納する。上記のように、これらのファイルは、ここでもプロキシサ

ーバの役割を果たすエージェントサーバ１５を介してダウンロードされる。このダウンロードは、これが終了するかまたはブラウザがTransition Sensorのstopイベントを生じさせるかのいずれか最初に起こるときまで続けられる。stopイベントについては、判断ブロック６６５はそれが生じるかどうかをテストし、このようなイベントがない場合、ＮＯ経路６６６を介て、実行は戻る。しかしながら、（上記のように）ユーザが開始するページ移行に応答し、このイベントが生じるときは常に、判断ブロック６６５はＹＥＳ経路６６８を介してブロック６７０へと実行する。この後者のブロックはこうして、メディアプレイヤー５６５を用いて、広告をプレイし、ブラウザのディスクキャッシュのプレイ待ち行列に完全に入れられる、すなわち、この広告についてのAdDescriptorファイルはプレイ待ち行列の先頭にあり、したがって、AdDescriptorファイルに特定されている、その広告に対する関連のメディアおよびプレイヤーファイルはすべてクライアントのハードディスクにある。

【０１１６】

３．AdControllerアプレット４２４

図７は、AdControllerアプレット４２４を実現する基本的な実行スレッドの高レベルブロック図を示す。

【０１１７】

図示のように、クライアントブラウザにより発生するTransition Sensorのinitイベントに応答して、あるスレッドがブロック７１０を実行し、AdControllerアプレット４２４を初期化する。このブロックでは、アプレット４２４のダウンロード（必要な程度だけ）およびインスタンス生成が行なわれる。クライアントブラウザにより発生するTransition Sensorのstartイベントに応答して、別のスレッドがブロック７２０の実行によってAdControllerアプレットを開始する。このアプレットが始まると、今度はこのアプレットが上記のように、ブロック７３０の実行を通して、広告（メディアおよびプレイヤー）ファイルのダウンロードの開始をイネーブルする。Transition Sensorアプレットが提供する遠隔広告管理システム（ここでは例として図１Ｂおよび１Ｃに示したシステム２５）の受信インターネットアドレスに応答して、第３のスレッドは、図７に示す実行ブロック７４０を通して、このアドレスにある広告管理システムからAdDescriptorファイルを要求し、これに応答して受けたAdDescriptorファイル６４５をダウンロードする。このときまでにブロック７３０で広告のダウンロードがイネーブルされている場合、AdDescriptorファイル６４５で特定されている広告ファイルは、必要に応じて「礼儀正しく」ダウンロードされる。クライアントブラウザにより発生し、隙間期間の開始を通知する、Transition Sensorのstopイベントに応答して、別のスレッドが、ここではブロック７５０の実行により、待ち行列に入れられた広告を表示するために、広告ファイルのダウンロードのサスペンドを開始する。このダウンロードがサスペンドされると、この最後のスレッドはブロック７６０を呼出し、このときプレイ待ち行列の先頭にある広告を、そのAdDescriptorファイルについてプレイを開始する。

【０１１８】

図８は、AdControllerアプレット４２４が実行する処理動作８００の高レベルフローチャートを示す。

【０１１９】

Transition Sensorアプレットにより発生するinitイベントに応答し、動作８００に入ると、ブロック８１０が行なわれる。このブロックを通して、AdControllerアプレットが初期化される。これは、必要な程度まで、エージェントサーバからこのアプレットについてのファイルをダウンロードし、このアプレットのインスタンス生成を行なうことを含む。これが発生すると、ブロック８１０では、Transition Sensorアプレットにより生じるAdControllerのstartイベントの発生についてテストを行なう。このイベントが生じるまで、ここの実行はＮＯ経路８１２を介してブロック８１０に戻るだけである。このイベントが生じると、判断ブロック８１０の実行はＹＥＳ経路８１４を介しブロック８１５に進む。この後者のブロックでは、エージェントサーバから外部AdController構成ファイル６２０

を取出す。その後、ブロック８２０が生じ、ここを通してAdControllerアプレットはAd Pipeline５４５を作成し開始する。このPipelineが完全に開始されると、ブロック８２５が実行され、広告ファイルがAd Pipelineに「礼儀正しく」ダウンロードされその後こういったファイルが実際にダウンロードされるようイネーブルする。広告ファイルがダウンロードされている間またはその後こういったダウンロードが完了した場合、判断ブロック８３０は、Play Adイベントの発生についてテストを行なう。こういったイベントが生じていない場合、実行は、ＮＯ経路８３３を介して判断ブロック８３０に戻り、さらなるダウンロードを続ける。しかしながら、Play Adイベントが生じた場合、判断ブロック８３０の実行は、ＹＥＳ経路８３７を介してブロック８４０に進む。この後者のブロックでは、Ad Pipelineへの広告ファイルのさらなるダウンロードをサスペンドする。これが発生すると、ブロック８４５が実行され、Ad Pipelineに対し、そのときプレイ待ち行列の先頭に位置する、AdDescriptorファイルを有する広告をプレイせよという要求を発行する。この広告がプレイされている間、判断ブロック８５０は、ブラウザが生じさせるシャットダウンイベントの発生についてテストを行なう。これはたとえば、ユーザが広告ウィンドウの移行を開始することまたはユーザが広告ウィンドウを閉じることによって、またはブラウザそのものを閉じることによって生じる。このようなイベントが発生しない場合、判断ブロック８５０は、ＮＯ経路８５３を介して実行をブロック８２５に戻し、広告ファイルの「礼儀正しい」ダウンロードを再びイネーブルする。このようなシャットダウンイベントが生じた場合、処理動作８００はＹＥＳ経路８５７を介して終了する。

【０１２０】

図９Ａおよび９Ｂは全体として、AdControllerアプレット４２４が具体的には広告を処理するために実行する処理動作９００のフローチャートを示し、この図の紙面の正しい配置を図９に示す。

【０１２１】

動作９００に入ると、ブロック９０５が実行され、Transition Sensorアプレットが発行する、次の広告、具体的には対応するAdDescriptorファイルをダウンロードせよという要求を受ける。この要求は、遠隔の広告管理システムのインターネットアドレスを含む。この要求に応答して、AdControllerアプレット４２４は、ブロック９１０を実行し、Ad Producerプロセス（これもスレッド）１５００に、広告をダウンロードするよう要求する。Ad Producerプロセスは、図１５と関連して以下で説明するように、Transition Sensorアプレットが送るインターネットアドレスから、具体的にはAdDescriptorファイルである広告ファイルを要求する。その後、ブロック９１５を通して、Ad Producerプロセスは、このプロセスが遠隔広告管理システムのインターネットアドレスを受けるまで、ブロックする（すなわち入力データを能動的に待つ）。その後、ブロック９２０の実行により、Ad Locationプロセス（これもスレッド）１６００は、AdDescriptorファイルが完全にAd Producerプロセス１５００によりダウンロードされAd Locationプロセスに与えられるときまで、ブロックする。図１６と関連して以下で説明するように、Ad Locationプロセス１６００は以下のタスクを実行する。（ａ）プロセス１６００の開始時に、このプロセスはAd Producerオブジェクトを作成する。（ｂ）Ad Producerプロセス１５００に次のAdDescriptorファイル６４５を要求する。（ｃ）プロセス１６００がこのようなAdDescriptorファイル６４５を得、ダウンロード待ち行列１４３０（図１４参照）が満杯でない場合、そのファイルをこの待ち行列に書込む。この待ち行列が満杯の場合、プロセス１６００は単に、その待ち行列が、AdDescriptorファイルをその待ち行列に書込む前に、満杯でないようになるまで待つ。AdDescriptorファイルが完全にダウンロードされると、Ad Locationプロセス１６００は、ブロック９２５で示すように、このファイルをダウンロード待ち行列１４３０に挿入する。

【０１２２】

AdDescriptorファイル６４５がダウンロード待ち行列に挿入されると、Ad Downloaderプロセス（これもスレッド）１７００が実行する。このプロセスは、以下で図１７と関連して説明するように、一続きのタスクを実行する。

【０１２３】
最初に、ブロック９３０で示すように、プロセス１７００は、そのときダウンロードすべき広告についてのAdDescriptorファイルがダウンロード待ち行列で利用可能になるまで、ブロックする。この実行中、このプロセスは、ダウンロード待ち行列１４３０に対し、そこにAdDescriptorファイルがあるかどうかすなわちダウンロードすべき広告ファイルについてのファイルがあるかどうかを尋ねる。ダウンロード待ち行列が空の場合、AdDescriptorプロセス１７００は、その待ち行列が空でなくなるまで待ち、かつ、ネットワークを通してAdDescriptorファイルを検索する。Ad Downloaderプロセスは、AdDescriptorファイルを発見すると、ブロック９４０で示すように、AdDescriptorファイルに特定された広告ファイルすべてを、ブラウザキャッシュプロキシ１４５０（図１４参照）を用いることによって、ブラウザのディスクキャッシュに（メディアファイルの場合はブラウザのＲＡＭキャッシュ）ダウンロードする。すべての広告ファイルのダウンロードが終了すると、Ad Downloaderプロセスは、ブロック９５０で示すように、AdDescriptorファイルをプレイ待ち行列１４７０に移動させる（図１４参照）。しかしながら、プレイ待ち行列がそのとき満杯ならば、Ad Downloaderプロセスは、AdDescriptorファイルをその待ち行列に動かす前に、プレイ待ち行列が満杯でなくなるなるまで待つ。

【０１２４】
ブラウザキャッシュプロキシは、抽象キャッシュに対するインターフェイスを実現する。このキャッシュの実現は、ブラウザのディスクまたはＲＡＭキャッシュ、Java仮想メモリキャッシュ、ローカルな生ディスクキャッシュなどといった種類のうちどんなものでもよい。このキャッシュプロキシを通過すると、広告を構成するメディアファイルは、ブラウザのディスクおよびＲＡＭキャッシュ双方にダウンロードされる。その後Ad Downloaderプロセスにおいてダウンロードされたものと同一のＵＲＬを有するメディアファイルにアクセスしようとするときは常に、このプロセスは最初に、インターネットを介してファイルをその広告管理サーバからダウンロードする代わりに、ブラウザのディスクキャッシュまたはブラウザのＲＡＭキャッシュからファイルをロードしようとする。このようにして、異なるウェブページまたはサイトを参照していても、可能な限り、異なる広告にまたがる広告ファイルの１回のダウンロードを強化する。

【０１２５】
次に、Transition Sensor stopイベントが生じるすなわち次の隙間期間の開始が示される場合、Transition Sensorのstopメソッド１８００は、AdControllerアプレット４２４が広告をプレイすることを要求する。この要求に応答して、アプレット内のイベントスケジューラスレッドはブロック９５５で示すように、アプレット４２４が広告のプレイを開始することによりこの要求に応答するまでブロックする。イベントスケジューラスレッドは、ユーザへの広告のプレイを制御する。このプロセスは、プレイ待ち行列の、次の広告に特有のメディアプレイヤーをいつ実行するかを判断し（すなわちその待ち行列にある、対応のAdDescriptorファイルについて）、かつ、プレイヤーが実行するコールバックメソッドを、そのプレイヤーが対応するAdDescriptorファイルに特定されている広告のプレゼンテーションをうまく終了したときに与える。AdControllerアプレットが広告のプレイを開始すると、ブロック９６５で示すように、イベントスケジューラは、そのときプレイ待ち行列の先頭にある広告、具体的には対応のAdDescriptorファイルを取出す。その後、イベントスケジューラは、ブロック９７０に示すように、対応するAdDescriptorファイルに特定されている特定のメディアプレイヤー５６５（図５参照）の実行を開始し、この特定の広告をプレイする。ブラウザのディスクキャッシュは、この広告について関連するコンテンツファイルをメディアプレイヤーに与える。広告の完全なプレゼンテーションの後、ブロック９７５で示すように、AdControllerアプレット４２４は適切に、このプレゼンテーションを、ブラウザのディスクキャッシュに保持されているログファイルに記録し、エージェントサーバへの次のアップロードに備える。こうして動作９００の実行は終了する。

【０１２６】
loggerプロセス（これもスレッドとして実現される）は、エージェントサーバに送り返す

べきすべてのログエントリを追跡し続ける。このプロセスでは単に、エントリの日時を記録し、これをログバッファに加える。次に、loggerプロセスは定期的に、こうしたエントリを保管し解析できるエージェントサーバへとログを戻しフラッシュする。

【０１２７】

広告について、プレイヤーメカニズムは、関連するAdDescriptorファイルに特定された関連のメディアファイルをブラウザキャッシュから取出し、これらのファイルを、見ることのできるフレームまたはウィンドウを介して実際にユーザに対して表示する。ユーザは、ブラウザのディスクキャッシュからの予めキャッシュされ円滑にプレイされている広告を眺める。その広告はメディアファイルに適している場合はブラウザＲＡＭキャッシュからの広告であり、インターネットを通してストリーミングされているものではない。広告表示のための４つのモードがサポートされている。すなわち、ユーザイベントによりトリガされた広告プレイ、フレームターゲット広告プレイ、タイマ型広告プレイおよびPopUp Javaフレームプレイである。これらのプレイヤーメカニズムは各々、メディアプレイヤーモジュール（図５に示すメディアプレイヤー５６５内に含まれる）およびプレイヤースレッドを利用する。このプレイヤースレッドにより、そのときクライアントブラウザを操作しているユーザに対する広告メディアの実際のプレゼンテーションが行なわれる。プレイヤーおよびプレイヤースレッドを組合せることにより、エージェントが構成できるタイマを用いた広告プレイの時間に基づく周波数の制御、ブラウザのウィンドウまたはJavaフレームにおける広告メディアファイルの表示、かなりの時間（通常はAdDescriptorファイルに特定される広告の長さ）の待機、および広告完了の際または広告がAdDescriptorファイルにおいて早期の終了を行なうように構成されている場合はユーザの要求に応じて広告を視覚的に終了させることが可能になる。

【０１２８】

フレームターゲットプレイは、広告メディアをブラウザのウィンドウに表示する。このようなプレイはユーザの要求に応じて中断および再開可能である。タイマ型広告プレイでは、連続的にループしている別のスレッドを用いて、プレイ待ち行列からAdDescriptorファイルを得、プレイヤーおよびプレイヤースレッドを用いてその広告を表示し、このシーケンスを繰返す前に特定の期間だけスリープする。タイマ型広告プレイもユーザの要求に応じて中断および再開可能である。このタイプの広告プレイの結果、ユーザは一定の間隔で配信される広告を定期的に見ることになり、ユーザが開始するイベントによってではない。PopUp Javaフレームプレイは、これも連続してループする別のスレッドであり、プレイ待ち行列からAdDescriptorファイルを得、ユーザが開始する移行が発生しているという通知を待ち、予め規定された期間だけブラウザ内のディスプレイウィンドウをポップアップし（「ポップアップ」ウィンドウ）、そのウィンドウに広告を提示し、このシーケンスを繰返す前にポップアップウィンドウを取除く。PopUp Javaプレイヤーの結果、ユーザは、あるウェブページと次のページとの間で移行するときは常に、連続する広告を各々予め規定された期間だけ（これはこういった広告各々についてAdDescriptorファイルで規定されているように広告によって異なる）見る。広告が完全にプレイされ、上記のようにたとえばタイマ型広告プレイを通した場合のように、AdDescriptorファイルにその広告の再生の命令がない場合、関連するAdDescriptorファイルは効果的にプレイ待ち行列から「外される」。

【０１２９】

特に、先に述べたように広告ファイルのダウンロードは、別の非同期スレッドを用いて、連続的に、バックグラウンドプロセスとして効果的に生じる。Transition Sensorセンサのstopメソッドは（具体的には図１８と関連して以下で説明するTransition Sensor stopメソッド１８００）、AdDescriptorエージェントに対しプレイイベントを発生する責任がある。このイベントは、エージェントに、ダウンロードされた広告をユーザに提示する機会を知らせる。このstopメソッドは、埋め込まれた広告タグを含むウェブページからユーザが出たときは常に、クライアントのブラウザによって自動的に呼出される。特に、このメソッドは、AdControllerエージェントのstart playerメソッドを呼出す。このstart pl

ayerメソッドは、今度は、イベントスケジューラにおいて同様に名づけられた、コンテンツページの移行中に広告のプレゼンテーションを開始および制御するメソッドを呼出す。イベントスケジューラは、広告に対するすべてのメディアファイルが、そのプレゼンテーションの前に、わからないようにダウンロードされることを保証し、また、広告を見えるように表示するのに必要な適切なプレイヤークラスの実際の実行を制御する。この点について、イベントスケジューラは、現在の広告に適切なプレイヤークラスについて、そのクラスのstartメソッドを呼出すことにより、インスタンス生成および呼出を行なう。このstartメソッドによって、この広告を視覚的に表示するプレイヤースレッドが生成される。次に、このstartメソッドは、ブラウザのディスクおよびＲＡＭキャッシュから広告メディアを視覚的に提示するために、プレイヤースレッドのrun（実行）メソッドを呼出す。完了後、広告の構成に基づいて、実行メソッドは、それ自身のstopメソッドを実行することにより、ユーザによるクローズ要求の検出の際に、または広告プレイのタイムアウトの完了時に、広告を終了させる。stopメソッドは、プレイヤーソフトウェア終了およびクリーンアップを行ない、最後はスケジューラオブジェクトをコールバックする。

【０１３０】

４．Transition Sensorアプレット４２２を含むアプレット間イベント

図１０は、Transition Sensorアプレット４２２の実行中にAdControllerエージェント４２０内で発生するアプレット間イベント１０００を示す。

【０１３１】

図示のようにかつ先に説明したように、ブラウザは広告タグ４０具体的にはコンテンツページ３５内にあるタグ４２を解釈および実行したときは常に、スクリプト２００（図２Ａおよび２Ｂ参照）をエージェントサーバからダウンロードする。今度は、このアプレットが動的にTransition Sensorアプレット２１０を、参照しているウェブコンテンツページに書込む。上記のように、このアプレットのインスタンス生成がクライアントのブラウザにより行なわれると、アプレットが今度は、アプレットレジストリ４２６のインスタンス生成を行なう。

【０１３２】

アプレットレジストリのインスタンス生成が行なわれると、Transition Sensorは、ライン１０１５で示す動作であるレジストリへの問合せを行ない、AdControllerアプレットの現在のステータスを判断する。ライン１０２０で示すように、このレジストリが、AdControllerアプレットがロードされておらずしたがって実行中でないことを示す場合、Transition Sensorアプレット４２２は、ライン１０２５で示すように、ブラウザのディスクキャッシュからAdControllerアプレット４２４をロードし、次にこのアプレットのインスタンス生成を行ないこれを開始する。AdControllerアプレットのインスタンス生成が行なわれると、Transition Sensorアプレットは、ライン１０３０で示すようにTransition SensorアプレットがロードされておりAdControllerアプレットがロードされていることを示す、適切なエントリを、アプレットレジストリに書込む。これが発生すると、アプレットレジストリは、ライン１０４０で示すように、AdControllerアプレットのための適切なハンドルをTransition Sensorに戻し、後者が前者を参照できるようにする。その後、ライン１０６０で示すように、Transition Sensorは、上記のように、広告管理システムのインターネットアドレスを含む要求をAdControllerアプレットに送り、広告に対するAdDescriptorファイルをそのアドレスからダウンロードする。このアドレスは広告タグ４０のタグ４４において特定されており、破線１０５０で示すように要求に組込まれている。その後、Transition Sensorは、ユーザが開始する次のコンテンツウェブページへの移行（クリックストリーム）に応答し、そのstopメソッド（図１８に示すメソッド１８００）を実行し、AdControllerアプレットに、そのときプレイ待ち行列の先頭にある、対応するAdDescriptorファイルを有する完全にダウンロードされた広告をプレイするよう指示する、すなわち、ライン１０６５で示すように要求を発行する。これが発生すると、Transition Sensorアプレットはその実行を、広告タグ４０を含むコンテンツページに次に出会い、これを解釈し実行するまで終了し、この時点でTransition Sen

sorアプレットは再びロードされ再び開始され以降同じように続けられる。

【０１３３】

５．Transition Sensorアプレット４２２

図１１は、Transition Sensorアプレット４２２を実現する基本的な処理スレッドの高レベルブロック図を示す。

【０１３４】

図示のように、クライアントブラウザが引き起こす、Init（初期化）Transition Sensorアプレットイベントに応答して、スレッドは、ブロック１１１０の実行により始まりTransition Sensorアプレット４２２を初期化する。このスレッドは今度、ブロック１１２０を実行してブラウザのディスクキャッシュからAdControllerアプレット４２４をロードするかまたはこれを必要であればエージェントのサーバからダウンロードし次にロードする。その後、このスレッドはブロック１１３０を実行して、広告タグ４０からのタグ４４における広告管理システム（たとえば図１Ｂおよび１Ｃ、２Ａおよび２Ｂならびに１０に示したシステム２５）のインターネットアドレスを得る。

【０１３５】

図１１に示すように、クライアントブラウザによる、Start Transition Sensorアプレットのstartイベントに応答して、別のスレッドがブロック１１４０の実行で始まり、Ad Downloaderプロセス１７００をイネーブルして（上記のようにかつ図１７と関連して以下でより詳細に説明するように）AdDescriptorファイルおよびすべての必要かつ関連の広告ファイル（メディアおよびプレイヤー双方）のブラウザディスクキャッシュへの「礼儀正しい」ダウンロードを開始する。

【０１３６】

さらに、図１１に示すように、クライアントブラウザによる、Stop Transition Sensorアプレットstopイベントに応答して、第３のスレッドが、ブロック１１５０の実行により始まり、Ad Downloaderプロセス１７００をディスエーブルして広告ファイルのさらなるダウンロードをサスペンドする。これが起こると、このスレッドはブロック１１６０を実行して、AdControllerアプレットに、そのときプレイ待ち行列の先頭にある、対応するAdDescriptorファイルを有する完全にダウンロードされた広告をプレイするよう指示する。

【０１３７】

図１２は、Transition Sensorアプレット４２２が実行する処理動作１２００の高レベルフローチャートを示す。

【０１３８】

動作１２００に入ると、判断ブロック１２１０は、クライアントブラウザによるinitイベントの発生についてテストを行なう。このようなイベントが発生するまで、実行はＮＯ経路１２１３を介してブロック１２１０に戻る。このイベントが生じると、実行がＹＥＳ経路１２１７を介してブロック１２２０へと続き、ここが実行されると、Transition Sensorアプレット４２２が初期化される。その後、ブロック１２３０が実行され、これを通してTransition Sensorアプレット４２４は、AdControllerアプレットに要求を発行することによって、広告、具体的には上記のように広告タグにおいて特定された広告管理サーバからのAdDescriptorファイルをダウンロードするよう指示する。これが発生すると、判断ブロック１２４０は、クライアントブラウザによりTransition Sensor startイベントが発生したかどうかをテストする。このようなイベントが発生するまで、実行は、ＮＯ経路１２４３を介してブロック１２４０に戻る。この特定のイベントが生じると、実行はＹＥＳ経路１２４７を介してブロック１２５０へと続き、ここでこれが実行されると、Ad Pipeline５４５がイネーブルされAdDescriptorファイルおよび関連の広告ファイルがダウンロードされる。

【０１３９】

次に、判断ブロック１２６０は、クライアントブラウザによりTransition Sensor stopイベントが生じたかどうかをテストする。このようなイベントが生じるまで、実行はＮＯ経路１２６３を介してブロック１２６０に戻る。Transition Sensor stopイベントが発生す

ると、実行はＹＥＳ経路１２６７を介してブロック１２７０へと続き、ここでこれが実行されると、AdControllerアプレット４２４に、具体的にはAd Pipeline５４５を介して広告をプレイするよう要求する。

【０１４０】

６．Ad Loaderプロセス１３００

図１３は、AdControllerアプレット４２４の一部を形成するAd Loaderプロセス１３００の高レベルブロック図を示す。プロセス１３００によって、広告主（具体的には広告プログラマ）に対し、AdControllerアプレットが実現する広告のプレイおよびログについての種々の機能にわたる制御を与える。具体的にはこのアプレットがいかにしてまたどこでネットワーク化された接続を通して広告を検索するのかおよびいかにしてこうした広告をプレイするかについての制御を行なう。Ad Loaderを用いることによって、外部からのプログラムによる呼出によりAdControllerアプレットを所望の程度制御できる。

【０１４１】

図示のように、このプロセスは、Ad Pipeline５４５へのインターフェイスを行なうAd Loader API（アプリケーションプログラミングインターフェイス）１３１０を含み、このパイプラインを通してプレイヤーメカニズムがブロック１３７０で示すようにいかにして広告を提示するかを制御する。特に、Ad Loader APIは、関連する情報を与え、種々のプログラム変数の設定を通して、プログラマが広告の表示およびダウンロード動作を制御できるようにする。この点について、こうした変数は、ユーザが移行したばかりのコンテンツページのダウンロードがいつ終了するかを示す、AdControllerアプレットへのコールバックをもたらし、これを用いてAdControllerアプレットに、いつ次の広告をダウンロードすべきなのか、完全にAd Pipelineの待ち行列に入れられた次の広告をプレイし、プレイタイマ（例として先に説明したようにタイマ型広告プレイに使用するために）をスタートおよびストップし、メッセージをログし、モードをセットして広告を表示するのに所望の場所を特定して、Ad Pipelineへの広告ファイルのダウンロードをサスペンドおよび再開し、所与の期間現在のダウンロードをサスペンドし、プレイヤーメカニズムにより広告プレイをサスペンドおよび再開するのか指示する。

【０１４２】

この点について、Ad Loader APIは、Ad Pipeline５４５を構成して、AdDescriptorファイル６４５がブロック１３２０で示すように遠隔広告管理システムから、広告管理システムのインターネットアドレス、および対象の広告については参照ウェブページアドレスのＵＲＬの受取に応答して、Ad Pipelineにダウンロードされる。ブロック１３３０で示すように、APIはAd Pipelineを構成して、広告のダウンロードが、AdControllerアプレット４２４が広告をプレイしていないときのみイネーブルされるようにする。さらに、ブロック１３４０で示すように、APIはAd Pipelineを構成して、広告のダウンロードがAdControllerアプレットが広告をプレイしているときは常にディスエーブルされるようにする。さらに、ブロック１３５０で示すように、APIはAd Pipelineを構成して、広告プレイが、次の広告をプレイせよという要求に応答して始められるようにする。すなわちブラウザのディスクキャッシュに完全にキャッシュされておりそのときプレイ待ち行列の先頭にあるAdDescriptorファイルを有する広告をプレイせよという要求である。

【０１４３】

７．Ad Pipeline５４５

図１４は、Ad Pipeline５４５の高レベルブロック図を示す。上記のように、Ad Pipelineは、集合的に、広告ファイル（必要なメディアおよびプレイヤーファイル）をブラウザのディスクキャッシュにロードし、メディアファイルについてはブラウザのＲＡＭキャッシュにもロードして、完全にダウンロードされた広告を提示する、種々のスレッドおよびデータ構造を実現する。上記のように、Ad Pipelineは、Ad Producerプロセス１５００、Ad Locationプロセス１６００およびAd Downloaderプロセス１７００（これらすべても上記のようにスレッドである）を用いる。

【０１４４】

入来する、広告をダウンロードせよという要求に応答して、Ad Pipeline５４５が呼出される。具体的には、このパイプライン内において、第１のブロック１４１０の実行により、広告のダウンロード要求の入来に応答してAd Producerプロセス１５００が呼出される。上記のように、Transition Sensorアプレットが発行するこの要求は、広告がありダウンロードされるべき（プロキシサーバとしてのエージェントサーバ１５を通して）、遠隔の広告管理システム（たとえば図１Ｂおよび１Ｃに示すシステム２５）のインターネットアドレスを含む。以下で図１５と関連して説明するように、Ad Producerプロセス１５００は、広告ファイルを要求する。具体的には、要求において特定されているインターネットアドレスからのAdDescriptorファイル（たとえばファイル６４５）である。この実行中、Ad Producerプロセスは、遠隔の広告管理システムのインターネットアドレスを受取るまで待機し、それからこのプロセスは特定された広告管理システムからAdDescriptorファイル６４５をダウンロードする。このファイルがダウンロードされると、図１４に示すブロック１４２０の実行により、Ad Locationプロセス１６００を呼出す（これについては以下で図１６と関連して詳細に説明する）。この実行中、Ad Locationプロセス１６００は、AdDescriptorファイル６４５がAd Producerプロセス１５００により完全にダウンロードされAd Locationプロセスに与えられるまでブロックし、その後、Ad Locationプロセスは、このAdDescriptorファイルをダウンロード待ち行列１４３０に書込む。

【０１４５】

AdDescriptorファイル６４５がダウンロード待ち行列に書込まれると、Ad Locationプロセス１６００は、以下で図１６と関連して説明するように、以下のタスクを実行する。（ａ）プロセス１６００の開始時にこのプロセスはAd Producerオブジェクトを作成する。（ｂ）このプロセスはAd Producerプロセス１５００に次のAdDescriptorファイル６４５を要求する。（ｃ）このプロセス１６００がAdDescriptorファイル６４５を入手し、ダウンロード待ち行列１４３０が満杯でない場合、プロセス１６００はそのファイルをこの待ち行列に書込む。この待ち行列が満杯の場合、プロセス１６００は単に、AdDescriptorファイルを待ち行列に書込む前に、待ち行列が満杯でないときまで待つ。AdDescriptorファイルが完全にダウンロードされると、Ad Locationプロセス１６００は、ブロック９２５で示すように、このファイルをダウンロード待ち行列１４３０に挿入する。

【０１４６】

AdDescriptorファイル６４５がダウンロード待ち行列に挿入されると、ブロック１４４０の実行により、Ad Downloaderプロセス１７００が呼出される。図１７と関連して以下で説明するプロセス１７００は、一続きのタスクを実行する。まず、プロセス１７００は、ダウンロードされたAdDescriptorファイルがダウンロード待ち行列で利用可能になるまでブロックする。この実行中、このプロセスは、ダウンロード待ち行列１４３０に、AdDescriptorファイルたとえばファイル６４５を含んでいるかどうか問合せる。そうであれば、広告ファイルをその特定のAdDescriptorファイルのためにダウンロードする必要がある。ダウンロード待ち行列が空の場合、プロセス１７００は、その待ち行列が空でなくなるまで待ちかつネットワークを通してAdDescriptorファイルを検索する。Ad Downloaderプロセス１７００がこのファイルを得ると、プロセス１７００は、ブラウザキャッシュプロキシ１４５０を用いることにより、AdDescriptorファイルにおいて特定されたすべてのメディアおよび必要なプレイヤーファイルを、ブラウザのディスク（およびＲＡＭ）キャッシュ１４６０にダウンロードする。広告ファイルすべてのダウンロードが完了すると、プロセス１７００はAdDescriptorファイルをプレイ待ち行列１４７０に移動させる。しかしながら、そのときプレイ待ち行列が満杯であれば、Ad Downloaderプロセスは、AdDescriptorファイルを後続の広告のプレイのためにこの待ち行列に移動させる前に、プレイ待ち行列１４７０が満杯でなくなるまで待つ。上記のように、完全に待ち行列に入れられた広告（すなわちクライアントのハードディスクにある関連のメディアおよびプレイヤーすべてを伴っている）に対するAdDescriptorファイル６４５は次に、広告をプレイせよという要求に応答してプレイ待ち行列１４７０から取出され、この要求はTransition Sensor stopイベントに応答して発行されるものである。

【０１４７】
８．Ad Producerプロセス１５００
図１５は、Ad Producerプロセス１５００の高レベルブロック図を示す。上記のように、このプロセスは、Transition Sensorアプレットが送るインターネットアドレスからAdDescriptorファイルを要求し、その後そのファイルをブラウザのディスクキャッシュにダウンロードする。
【０１４８】
図示のように、プロセス１５００に入ると、ここでの実行はまず判断ブロック１５１０へと進む。このブロックでは、ＵＲＬがTransition Sensorから受取られたかどうかを判断し、ここからAdDescriptorファイルがフェッチされる。このようなＵＲＬがまだ受取られていない場合、実行はＮＯ経路１５１７を介してこの判断ブロックに戻る。その代わりに、このようなＵＲＬが受取られていれば、実行はＹＥＳ経路１５１３を介してブロック１５２０へと進み、ここではこのＵＲＬを広告ＵＲＬ１５３０として、次に続く広告ダウンロードの機会中に用いるために格納する。
【０１４９】
このＵＲＬがこのようにして格納されると、実行は判断ブロック１５４０に進む。このブロックは、ユーザが開始するイベント（クリックストリーム）の発生についてテストする。これは、たとえばユーザが現在の広告フレームを閉じたときなどに広告のダウンロードが可能になり、ユーザが移行した次に続くコンテンツページがクライアントブラウザにより表示されていることを示す。このようなイベントが発生していなければ、たとえば次に続くコンテンツウェブページがダウンロードされていれば、実行はただＮＯ経路１５４３を介して判断ブロック１５４０に戻る。しかしながら、このようなイベントが発生していれば、この判断ブロックはＹＥＳ経路１５４７を介して実行をブロック１５５０へと進める。この後者のブロックが実行されると、AdDescriptorファイル６４５が、Transition Sensorが送るＵＲＬを用いてダウンロードされる。このファイルが完全にダウンロードされると、ブロック１５６０の実行によりこのファイルがAd Locationプロセス１６００に転送される。その後、実行は経路１５６５を介して判断ブロック１５１０へと戻り、同じように続けられる。
【０１５０】
９．Ad Locationプロセス１６００
図１６は、Ad Locationプロセス１６００の高レベルブロック図を示す。このプロセスは上記のように、以下のタスクを行なう。（ａ）このプロセスの開始時に、プロセス１６００はAd Producerオブジェクトを作成する。（ｂ）プロセス１６００は、Ad Producerプロセス１５００に次のAdDescriptorファイル６４５を要求する。（ｃ）プロセス１６００がAdDescriptorファイル６４５を得、ダウンロード待ち行列１４３０（図１４参照）が満杯でない場合、プロセス１６００はそのファイルをこの待ち行列に書込む。この待ち行列が満杯の場合、プロセス１６００は単に、AdDescriptorファイルを待ち行列に書込む前にこの待ち行列が満杯でないときまで待つ。
【０１５１】
プロセス１６００に入ると、広告のダウンロードそのものに関し、実行は判断ブロック１６１０へと進む。この判断ブロックが実行されると、広告管理システムのインターネットアドレス（ＵＲＬ）を次に続く広告のダウンロードのためにTransition Sensorアプレットから受けたかどうかを判断する。このアドレスをまだ受けていなければ、実行は単にＮＯ経路１６１３を介して判断ブロック１６１０に戻る。その代わりとして、このようなアドレスを受けたが未処理の場合、判断ブロック１６１０は、ＹＥＳ経路１６１７を介して実行をブロック１６２０へと進める。この後者のブロックは、Ad Producerプロセス１５００に、このＵＲＬからAdDescriptorファイルたとえばファイル６４５をダウンロードするよう求める。この要求が生じると、実行は判断ブロック１６３０へと進み、このAdDescriptorファイルが完全にダウンロードされているかどうかを判断する。このファイルのダウンロードがまだ生じている場合、実行は単にＮＯ経路１６３３を介してブロック１６３

０へと戻り、ダウンロードの完了を待つ。このダウンロードが完了すると、判断ブロック１６３０はＹＥＳ経路１６３７を介してブロック１６４０へと実行を進める。この後者のブロックでは、ダウンロードされたAdDescriptorファイルをダウンロード待ち行列１４３０に書込む（この待ち行列が一杯ではないとする）。これが発生すると、実行は経路１６４５を介し判断ブロック１６１０へと導かれ、以降同様に続く。

【０１５２】

１０．Ad Downloaderプロセス１７００

図１７は、Ad Downloaderプロセス１７００の高レベルブロック図を示す。本質的には、上記のように、プロセス１７００は、ダウンロード待ち行列から、その待ち行列がたとえばファイル６４５といったAdDescriptorファイルを含むかどうかを判断する。このようなAdDescriptorファイルが含まれていなければ、そのファイルについて広告ファイルをダウンロードする必要がある。結果的に、プロセス１７００ではそのAdDescriptorファイルに特定されている必要な広告ファイルをダウンロードする。これが十分に発生すると、プロセス１７００はAdDescriptorファイルをプレイ待ち行列に移動させる。

【０１５３】

特に、プロセス１７００に入ると、実行は判断ブロック１７１０へと進む。この判断ブロックは、ダウンロード待ち行列がそのときたとえばファイル６４５といったAdDescriptorファイルを含むかどうかを判断する。この待ち行列が空の場合、実行は単にＮＯ経路１７１７を介してこの判断ブロックに戻ってこのようなAdDescriptorファイルを待つ。しかしながら、ダウンロード待ち行列１４３０がこのようなファイルを含む場合、プロセス１７２０は、そのときこの待ち行列の先頭にあるAdDescriptorファイルを得る。その後、ブロック１７３０が実行される。このブロックでは、そのときにはクライアントのハードディスクにない、必要な広告ファイルすべてをブラウザのプロキシキャッシュ１４５０にダウンロードする。このブロックでもまた、ブラウザのプロキシキャッシュにある関連のメディアファイルすべてをブラウザのＲＡＭキャッシュに転送する。実行は次に判断ブロック１７４０へと進み、ここでは必要な広告ファイルすべてがダウンロードされているかどうかを判断する。このようなファイルのダウンロードが残っていれば、判断ブロック１７４０は実行をＮＯ経路１７４７を介してブロック１７３０へと戻し、そのファイルをダウンロードする。その代わりとして、必要な広告ファイルすべてがダウンロードされていれば、実行はＹＥＳ経路１７４３を介してブロック１７５０へと進む。この後者のブロックは、AdDescriptorファイルを、ダウンロード待ち行列１４３０からプレイ待ち行列１４７０の末尾へと移動させる。AdDescriptorファイルがプレイ待ち行列に書込まれると、対応する広告を、続く隙間期間中、そのときプレイ待ち行列に入れられている他のAdDescriptorファイルに関した順序で、ユーザに提示することができる。

【０１５４】

１１．Transition Sensor stopメソッド１８００

図１８は、Transition Sensorアプレット４２２が呼出すstopメソッド１８００のフローチャートである。この方法は、ブラウザが発生するstopイベントに応答して、広告ファイルのダウンロードをサスペンドして隙間広告プレイを開始する。

【０１５５】

特に、方法１８００に入ると、判断ブロック１８１０の実行により、stopイベントをブラウザ７から受けたかどうかを判断する。このようなstopイベントがまだ生じていない場合、実行はＮＯ経路１８１３を介してブロック１８１０に戻り、このイベントの発生を待つ。このイベントが発生すると、判断ブロック１８１０はＹＥＳ経路１８１７を介して判断ブロック１８２０へと実行を進める。この後者の判断ブロックでは、AdControllerアプレット４２４がそのときロードされ実行しているかどうかを判断する。このアプレットがそのとき実行していない場合、判断ブロック１８２０はＮＯ経路１８２７を介して実行をブロック１８３０に導く。この後者のブロックでは、AdControllerアプレットに対して広告をプレイせよと要求することを、このアプレットが実行するまで禁止し、これが生じると、次のユーザ開始（クリックストリーム）イベントが発生する。その後、方法１８００の

実行が終了する。その代わりとして、AdControllerアプレットがロードされ実行している場合、判断ブロック１８２０はＹＥＳ経路１８２３を介して実行をブロック１８４０に導く。この後者のブロックは、AdControllerアプレットに次の広告をプレイするよう要求する。この要求が発行されると、実行はブロック１８５０に進む。今度はこのブロックは、AdControllerアプレットに、広告ファイルの「礼儀正しい」バックグラウンドダウンロードをサスペンドするよう要求し、一方では次に続くウェブコンテンツページがユーザの要求に応じてブラウザからダウンロードされている。ブロック１８５０の実行により、方法１８００の実行は終了する。

【０１５６】

１２．Transition Sensor startメソッド１９００

図１９は、Transition Sensorアプレット４２２が呼出すstartメソッド１９００のフローチャートを示す。このメソッドは、ブラウザが発生するstartイベントに応答して、広告ファイルのバックグラウンドダウンロードを再開する。

【０１５７】

具体的には、方法１９００に入ると、実行は判断ブロック１９１０へと進み、実行されると、startイベントをブラウザ７が受けたかどうかが判断される。このようなstartイベントがまだ発生していない場合、実行はＮＯ経路１９１３を介してブロック１９１０に戻り、このイベントの発生を待つ。このイベントが発生すると、判断ブロック１９１０は実行をＹＥＳ経路１９１７を介して判断ブロック１９２０へと導く。この後者の判断ブロックでは、AdControllerアプレット４２４がそのときロードされ実行しているかどうか判断する。このアプレットが実行していなければ、判断ブロック１９２０は実行をＮＯ経路１９２７を介してブロック１９３０へと導く。ブロック１９３０では、AdControllerアプレットに対して広告をダウンロードせよと要求することを、そのアプレットが実行するまで禁止し、これが生じると、次のユーザ開始（クリックストリーム）イベントが発生する。AdControllerアプレットが実行を開始しその後次のユーザ開始（クリックストリーム）イベントが発生すると、実行はブロック１９４０に進む。この後者のブロックでは、AdControllerアプレットに、広告ファイルのバックグラウンドダウンロードを再開するよう求める。このダウンロードが再開されると、メソッド１９００は、ブロック１９６０の実行を通して、ユーザがブラウザが現在表示しているウェブページを次にアンロードするとき、すなわち、ユーザが開始するイベントを次に続くウェブページに移行させるときは常に、ブラウザ７がTransition Sensor stopメソッド１８００を呼出すのを待つ。その代わりとして、AdControllerアプレットがロードされ実行しているならば、判断ブロック１９２０は実行をＹＥＳ経路１９２３を介してブロック１９５０へと導く。この時点で次に続くコンテンツウェブページはブラウザによって完全に実行されており、かつ、たとえばユーザに対して表示されているため、ブロック１９５０では、アプレットレジストリを通してAdControllerアプレットに、広告ファイルのバックグラウンドダウンロードの再開をイネーブルするよう要求する。これが発生すると、ブロック１９４０が実行されバックグラウンドダウンロードの再開をAdControllerアプレットに要求する。実行は次にブロック１９６０に進み、ユーザが現在ブラウザが表示しているウェブページを次にアンロードするとき、すなわち、ユーザが開始するイベントが次に続くウェブページに移行させるときは常に、ブラウザ７がTransition Sensor stopメソッド１８００を呼出すのを待つ。ブラウザが次のTransition Sensor stopイベントを発生するときは常に、プロセス１９００は終了する。

【０１５８】

本発明の教示を組入れた１つの実施例が、本明細書において示されかつかなり詳細に説明されたが、当業者であれば、こうした教示を用いた、本願の多くの他の実施例および応用例を、容易に考案することが可能である。

【図面の簡単な説明】

【図１】　図１Ａは、図１Ｂおよび１Ｃの図の紙面の正しい配置を示す。図１Ｂおよび１Ｃは全体として、本発明の教示を実施する、インターネットを通して実現される例示のク

ライアントーサーバ配信処理環境の高レベルブロック図を示し、それとともに、本発明に関連するものとして、その環境で生じる基本的なコンピュータ間動作および関連のクライアント処理動作を示す。図１Ｄは、図１Ｅおよび１Ｆの図の紙面の正しい配置を示す。図１Ｅおよび１Ｆは全体として、図１Ｂおよび１Ｃに示したのと同じ環境を示しているが、この図面ではエージェントのダウンロード／インスタンス生成／実行動作５０を詳細に示している。

【図２】　図２Ａおよび２Ｂの図の紙面の正しい配置を示す。図２Ａおよび２Ｂは全体として、ウェブページのＨＴＭＬコード３５を一般化したものを示しており、具体的にはわからないように本発明を呼出す広告タグ４０を含み、さらに、そのコードに対して本発明が動的に行なう変更を示し、具体的にはウェブ広告のダウンロードおよび表示のために、Transition Sensorアプレット２１０をタグ４０と置き換えてページ３５′を作成している。

【図３】　図１Ｂおよび１Ｃならびに図１Ｅおよび１Ｆに示したクライアントＰＣ５の高レベルブロック図を示す。

【図４】　図３に示したクライアントＰＣ５内にあるアプリケーションプログラム４００の高レベルブロック図を簡単にしたものを示す。

【図５】　本発明を実現する、図４に示したAdControllerエージェント４２０の高レベルブロック図を示す。

【図６】　図６Ａおよび６Ｂの図の紙面の正しい配置を示す。図６Ａおよび６Ｂは全体として、図５に示したAdControllerエージェント４２０が実施する処理動作６００の高レベルフローチャートを示す。

【図７】　図４に示すように、AdControllerエージェント４２０の一部を形成するAdControllerアプレット４２４を実現する、基本的な処理スレッドの高レベルブロック図を示す。

【図８】　図７に示したAdControllerアプレット４２４が実行する処理動作８００の高レベルフローチャートを示す。

【図９】　図９Ａおよび９Ｂの図の紙面の正しい配置を示す。図９Ａおよび９Ｂは全体として、具体的には広告を処理するための、図７に示したAdControllerアプレット４２４が実施する処理動作９００のフローチャートを示す。

【図１０】　Transition Sensorアプレット４２２の実行中にAdControllerエージェント４２０内で生じるアプレット間イベントを示す。

【図１１】　AdControllerエージェント４２０の一部を形成する、図４に示した、Transition Sensorアプレット４２２を実現する基本的な処理スレッドの高レベルブロック図を示す。

【図１２】　図１１に示したTransition Sensorアプレット４２２が実行する処理動作１２００の高レベルフローチャートを示す。

【図１３】　AdControllerアプレット４２４が実現する広告のプレイおよびログのために、種々の機能にわたる広告主制御を行なうのに用いることができるAd Loaderプロセス１３００の高レベルブロック図を示す。

【図１４】　図４に示したAdControllerアプレット４２４が実行しかつこの一部を形成するAd Pipeline５４５の高レベルブロック図を示す。

【図１５】　図１４に示したAd Pipeline５４５が実行するAd Producerプロセス１５００の高レベルブロック図を示す。

【図１６】　図１４に示したAd Pipeline５４５が実行するAd Locationプロセス１６００の高レベルブロック図を示す。

【図１７】　図１４に示したAd Pipeline５４５が実行するAd Downloaderプロセス１７００の高レベルブロック図を示す。

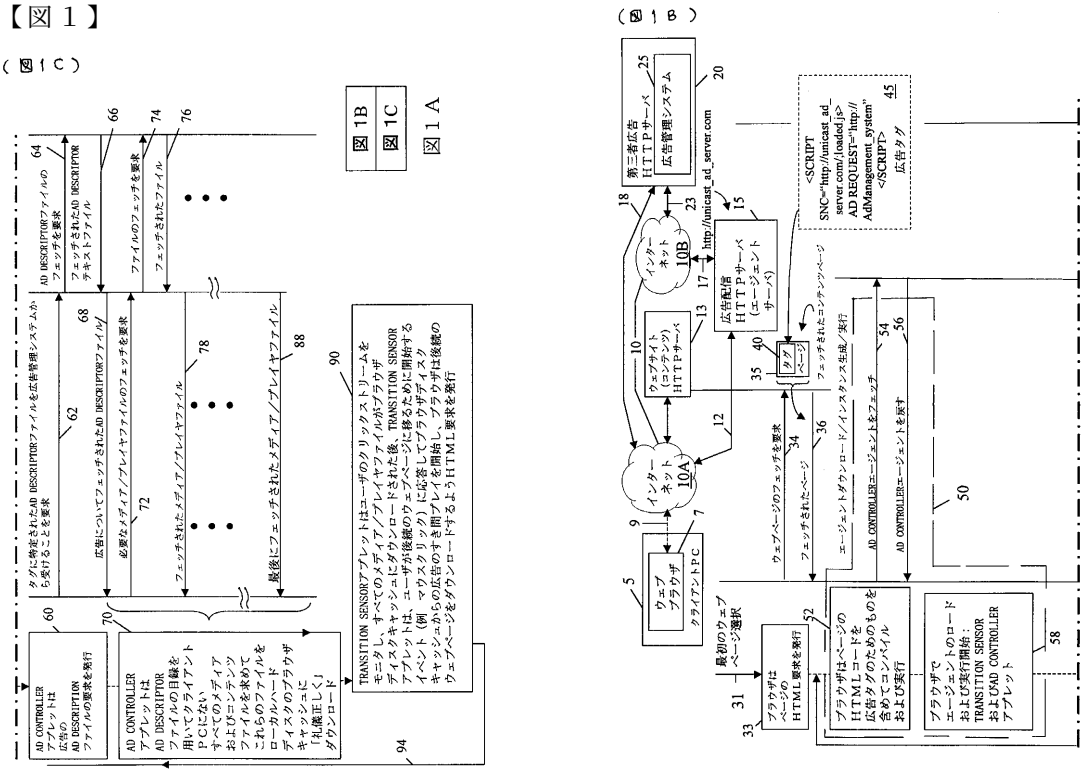
【図１８】　図４に示したTransition Sensorアプレット４２２が呼出すstopメソッド１８００のフローチャートを示す。

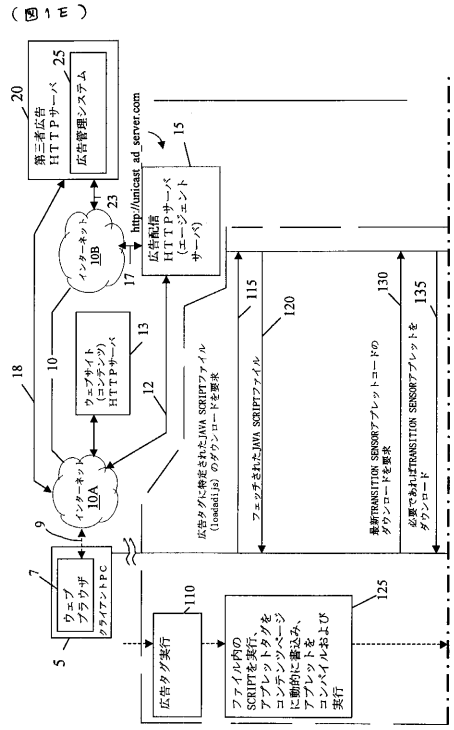
【図１９】　図４に示したTransition Sensorアプレット４２２が呼出すstartメソッド１

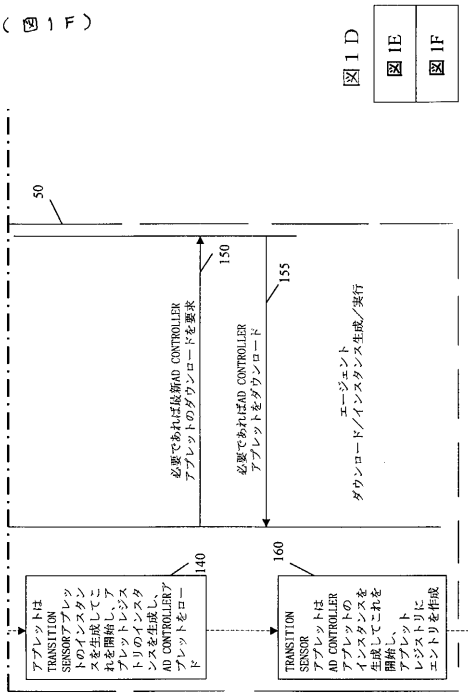
９００のフローチャートを示す。

【図２０】　本発明を通してPointCastタイプのJava広告を隙間で表示するのに用いるための、実際のAdDescriptorファイル２０００のコンテンツを例示したものである。

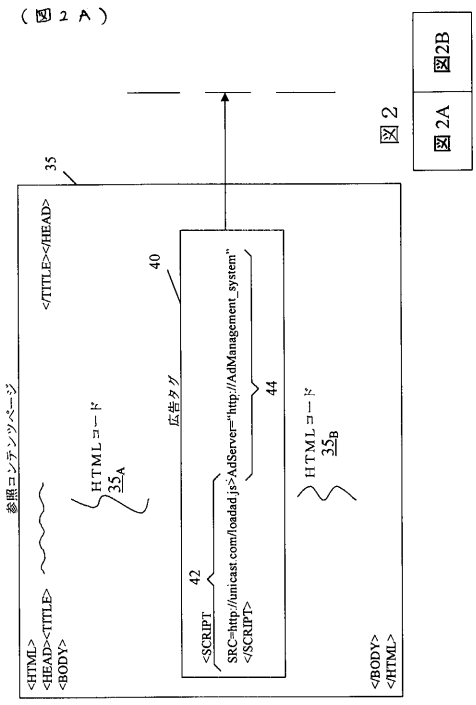
【図１】

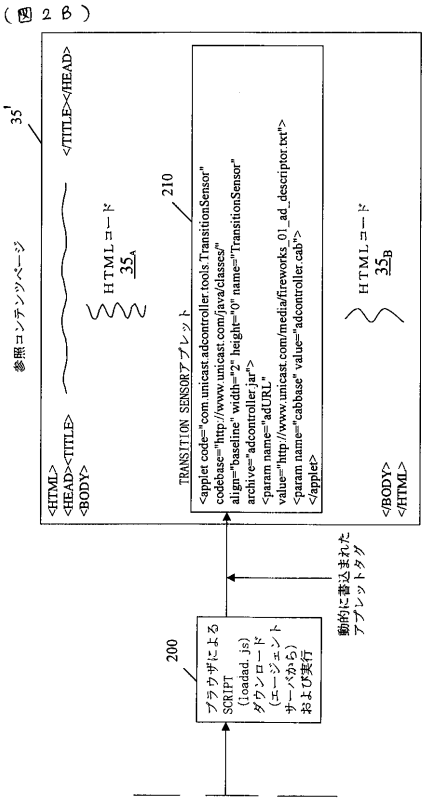



(図1E)
第三者広告HTTPサーバ
広告管理システム
http://unicast_ad_server.com
広告配信HTTPサーバ(エージェントサーバ)
インターネット10B
ウェブサイト(コンテンツ)HTTPサーバ
インターネット10A
ウェブブラウザクライアントPC
広告タグに特定されたJAVA SCRIPTファイル(loadad.js)のダウンロードを要求
フェッチされたJAVA SCRIPTファイル
最新TRANSITION SENSORアプレットコードのダウンロードを要求
必要であればTRANSITION SENSORアプレットをダウンロード
広告タグ実行
ファイル内のSCRIPTを実行、アプレットタグをコンテンツページに動的に書込み、アプレットをコンパイルおよび実行

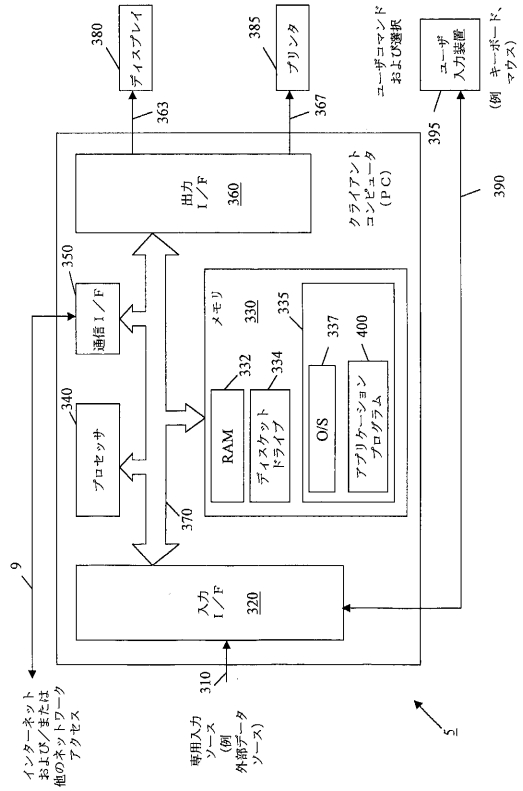

(図1F)
図1D
図1E
図1F
必要であれば最新AD CONTROLLERアプレットのダウンロードを要求
必要であればAD CONTROLLERアプレットをダウンロード
エージェントダウンロード/インスタンス生成/実行
アプレットはTRANSITION SENSORアプレットのインスタンスを生成してこれを開始しアプレットレジストリのインスタンスを生成しAD CONTROLLERアプレットをロード
TRANSITION SENSORアプレットはAD CONTROLLERアプレットのインスタンスを生成してこれを開始しアプレットレジストリにエントリを作成

【図2】

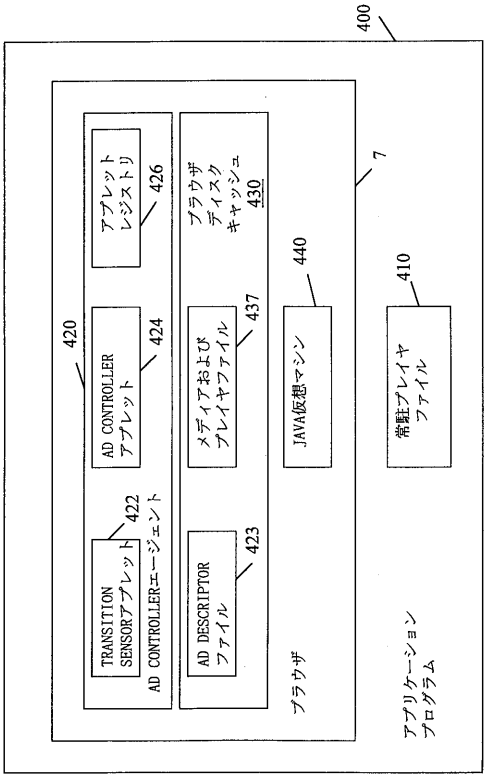

(図2A)
図2
図2A
図2B
参照コンテンツページ
HTMLコード 35A
広告タグ
SCRIPT
SRC=http://unicast.com/loadad.js?AdServer="http://AdManagement_system"
</SCRIPT>
HTMLコード 35B


(図2B)
参照コンテンツページ
HTMLコード 35A
TRANSITION SENSORアプレット
HTMLコード 35B
ブラウザによるSCRIPT(loadad.js)ダウンロード(エージェントサーバから)および実行
動的に書込まれたアプレットタグ

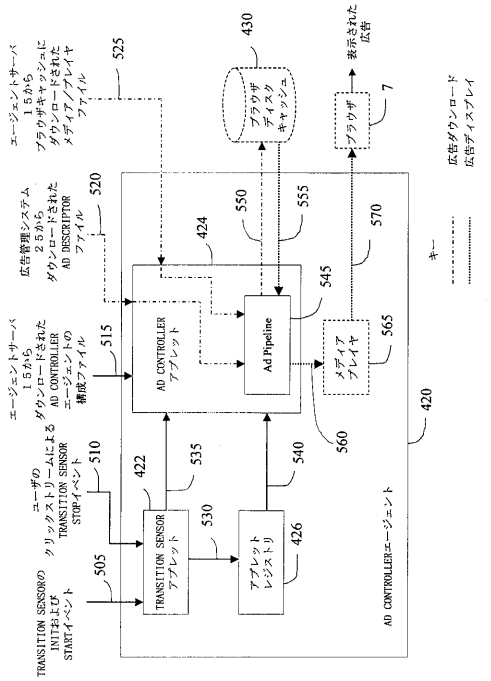
【図３】



【図４】

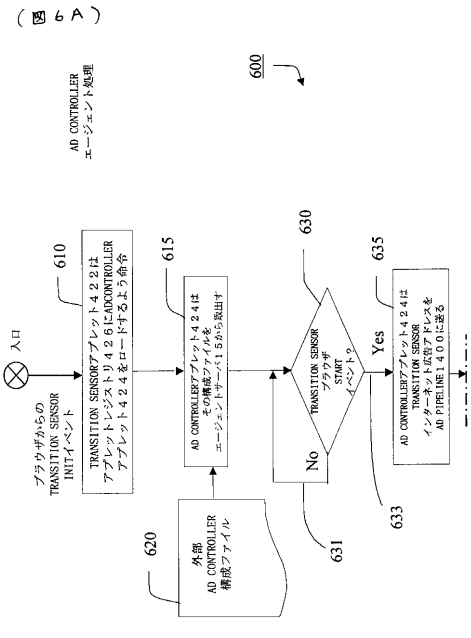


【図５】



【図６】

（図６Ａ）



（図６Ｂ）

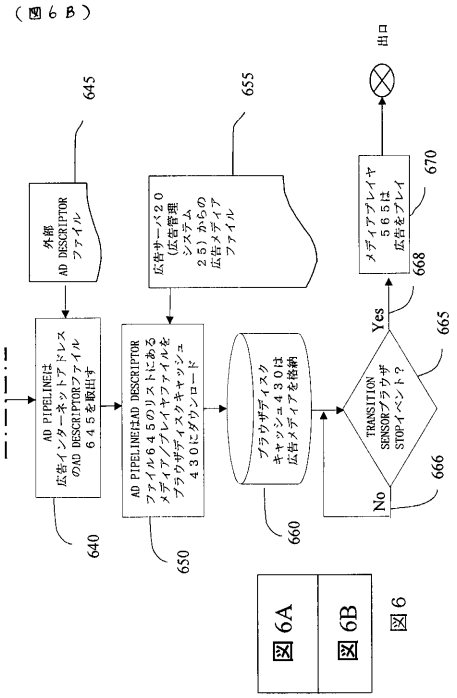
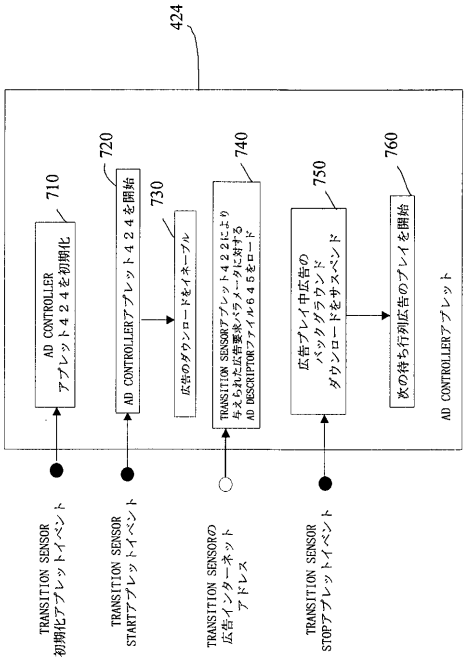


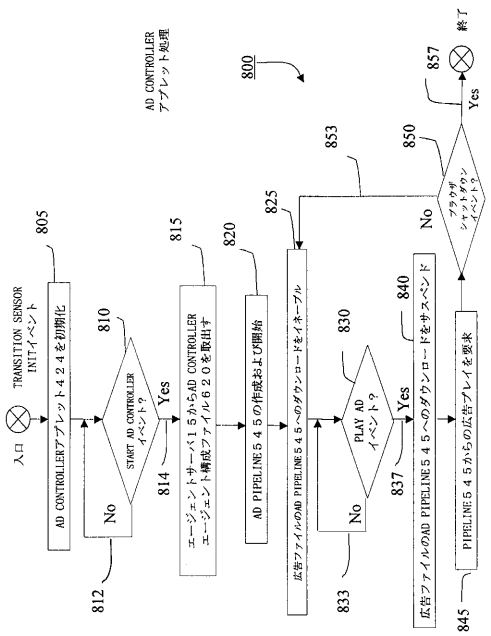
図６Ａ　図６Ｂ

図６

【図７】

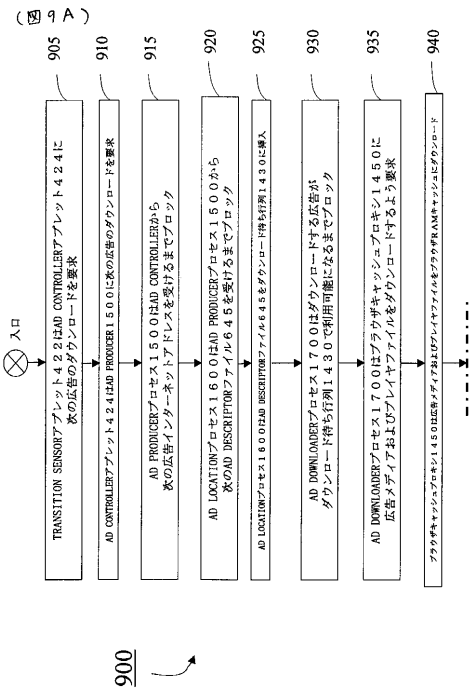


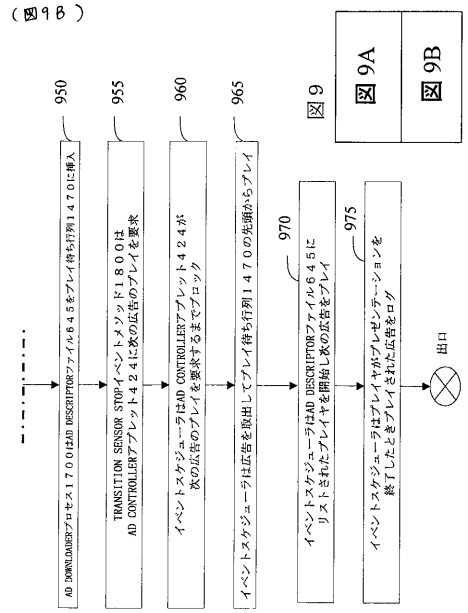
【図８】



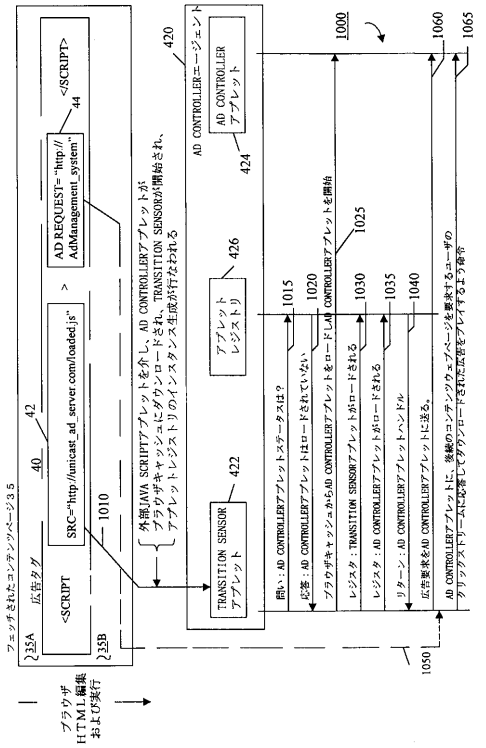
【図９】

（図９Ａ）

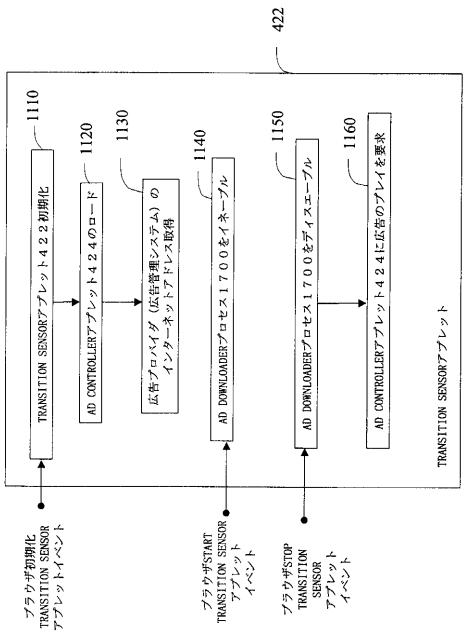



(図9B)
AD DOWNLOADERプロセス1700はAD DESCRIPTORファイル645をプレイ待ち行列1470に挿入
950
TRANSITION SENSOR STOPイベントメソッド1800は
AD CONTROLLERアプレット424に次の広告のプレイを要求
955
イベントスケジューラはAD CONTROLLERアプレット424が
次の広告のプレイを要求するまでブロック
960
イベントスケジューラは広告を取出してプレイ待ち行列1470の先頭からプレイ
965
970
イベントスケジューラはAD DESCRIPTORファイル645に
リストされたプレイヤを開始し次の広告をプレイ
975
イベントスケジューラはプレイヤがプレゼンテーションを
終了したときプレイされた広告をログ
出口
図9
図9A
図9B

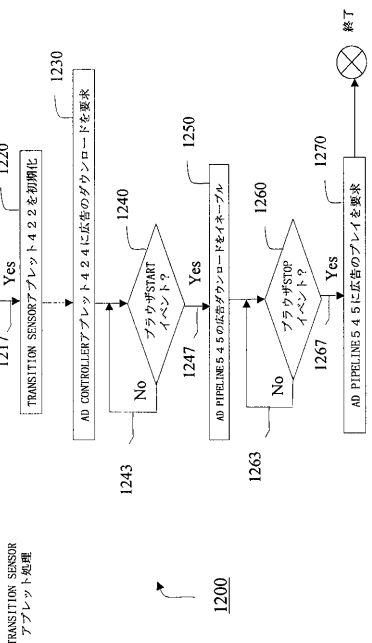
【図１０】


フェッチされたコンテンツページ35
広告タグ
35A
<SCRIPT
SRC="http://unicast_ad_server.com/loaded.js">
35B
40
42
1010
ブラウザHTML構築および実行
外部JAVA SCRIPTアプレットを介し、AD CONTROLLERアプレットが
ブラウザキャッシュにダウンロードされ、TRANSITION SENSORが開始され、
アプレットレジストリのインスタンスが生成が行なわれる
AD REQUEST="http://AdManagement_system"
<SCRIPT>
44
AD CONTROLLERエージェント
420
TRANSITION SENSOR アプレット
422
アプレットレジストリ
426
AD CONTROLLER アプレット
424
1000
問い：AD CONTROLLERアプレットステータスは？
1015
応答：AD CONTROLLERアプレットはロードされていない
1020
ブラウザキャッシュからAD CONTROLLERアプレットをロードFLAG CONTROLLERアプレットを開始
1025
レジスタ：TRANSITION SENSORアプレットがロードされる
1030
レジスタ：AD CONTROLLERアプレットがロードされる
1035
リターン：AD CONTROLLERアプレットハンドル
1040
広告要求をAD CONTROLLERアプレットに送る。
1050
AD CONTROLLERアプレットに、後続のコンテンツウェブページを要求するユーザの
クリックストリームに応答してダウンロードされた広告をプレイするよう命令
1060
1065

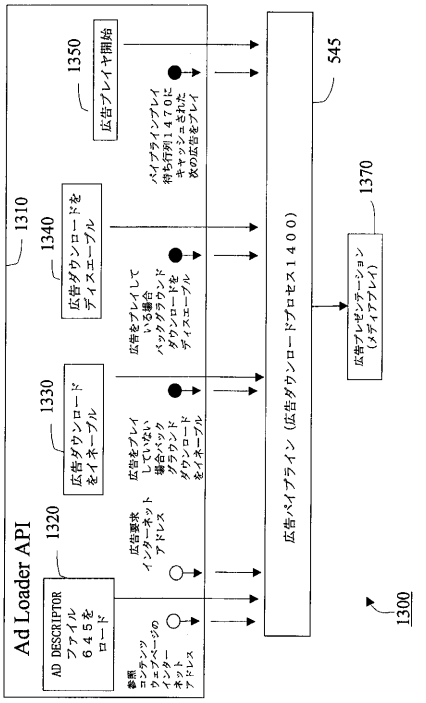
【図１１】


422
TRANSITION SENSORアプレット
TRANSITION SENSORアプレット422初期化
1110
AD CONTROLLERアプレット424のロード
1120
広告プロバイダ（広告管理システム）の
インターネットアドレス取得
1130
AD DOWNLOADERプロセス1700をイネーブル
1140
AD DOWNLOADERプロセス1700をディスエーブル
1150
AD CONTROLLERアプレット424に広告のプレイを要求
1160
ブラウザ初期化
TRANSITION SENSOR
アプレットイベント
ブラウザSTART
TRANSITION SENSOR
アプレット
イベント
ブラウザSTOP
TRANSITION
SENSOR
アプレット
イベント

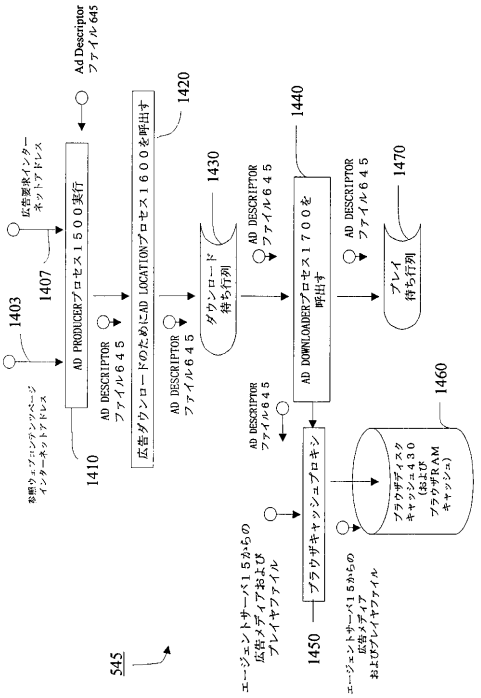
【図１２】


TRANSITION SENSOR
アプレット処理
1200
入口
ブラウザINITイベント？
1210
No
1213
Yes
1217
TRANSITION SENSORアプレット422を初期化
1220
AD CONTROLLERアプレット424に広告のダウンロードを要求
1230
ブラウザSTARTイベント？
1240
No
1243
Yes
1247
AD PIPELINE545の広告ダウンロードをイネーブル
1250
ブラウザSTOPイベント？
1260
No
1263
Yes
1267
AD PIPELINE545に広告のプレイを要求
1270
終了

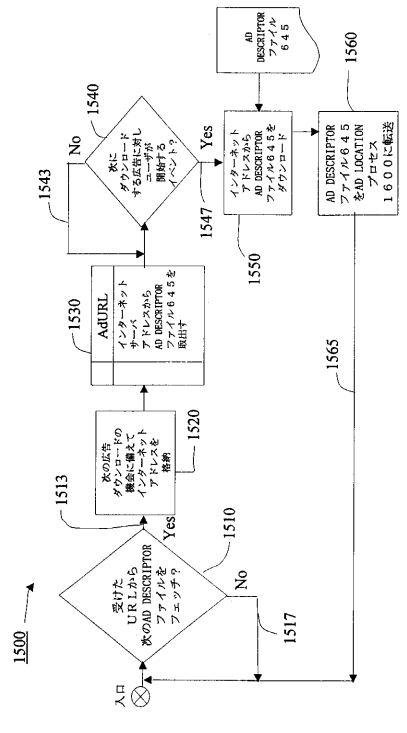
【図１３】



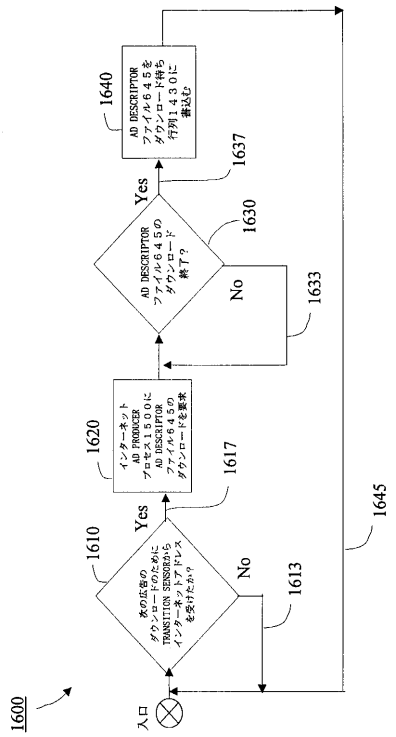
【図１４】



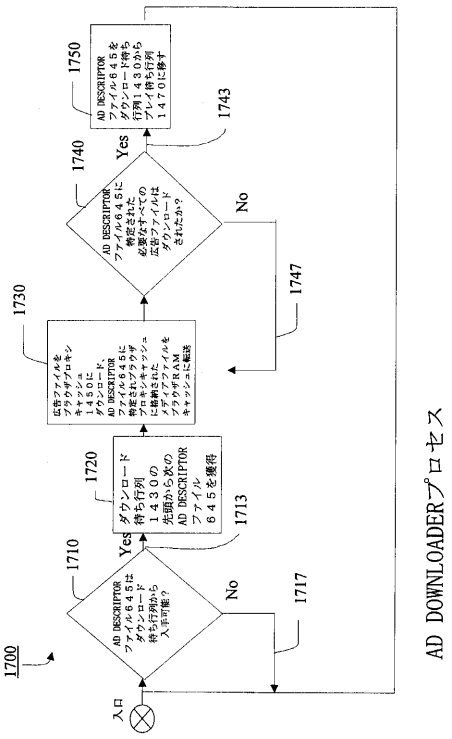
【図１５】



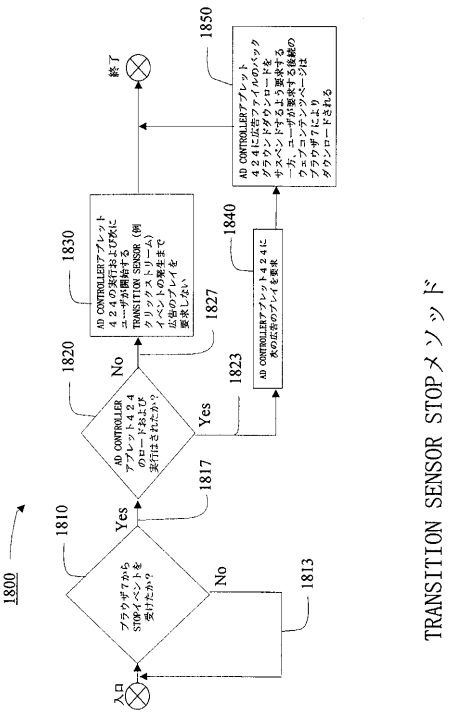
【図１６】



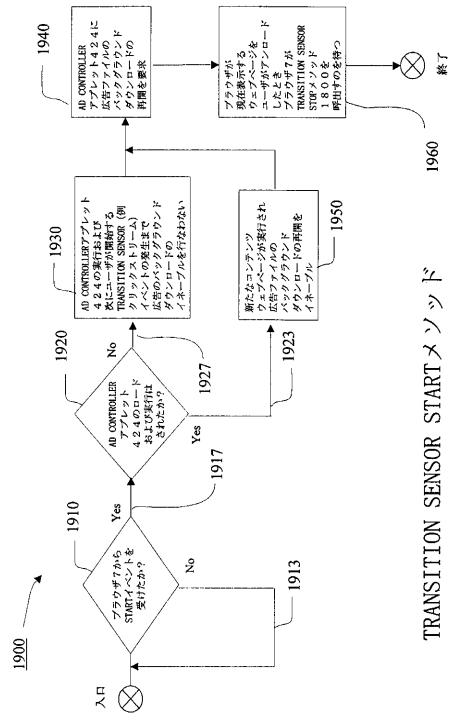
【図１７】



【図１８】



【図１９】



【図２０】

```
# Section 1 - Applet Player Java Class Configuration
playerName=AppletViewerPopup
playerClass=com.unicast.adcontroller.players.AppletViewerPopup
# Section 2 - Ad Java Classes Configuration
AppletViewerPopup.adAppletName=ANM_AnimationLoaderApplet
AppletViewerPopup.adAppletClass=
com.pointcast.applets.AnimationApplet.ANM_AnimationLoaderApplet
# Section 3 - Player Execution Configuration
AppletViewerPopup.windowTitle=AdController PointCast Ad
AppletViewerPopup.playerRefreshRate=1000
AppletViewerPopup.allowExit=true
AppletViewerPopup.xPosition=50
AppletViewerPopup.yPosition=50
AppletViewerPopup.windowWidth=280
AppletViewerPopup.windowHeight=355
AppletViewerPopup.isResizable=false
AppletViewerPopup.secondsWindowIsOpen=180
AppletViewerPopup.secondsToOverlay=1
AppletViewerPopup.closeButtonLabel=Close
AppletViewerPopup.openButtonLabel=More Info
AppletViewerPopup.saveButtonLabel=Save
AppletViewerPopup.openURL=http://www.pointcast.com/
# Section 4 - Ad Applet Configuration
# A. Ad Applet DocumentBase
AppletViewerPopup.ANM_AnimationLoaderApplet.documentBase=
http://www2.unicast.com/~rlandsma/AdController/MacromediaApplet/
# B. Ad Applet Parameters
AppletViewerPopup.ANM_AnimationLoaderApplet.AdToPlay=deepsea.anm
AppletViewerPopup.ANM_AnimationLoaderApplet.AltImage=test.gif
AppletViewerPopup.ANM_AnimationLoaderApplet.MaxCycles=5
AppletViewerPopup.ANM_AnimationLoaderApplet.TargetURL=http://www.pointcast.com/
AppletViewerPopup.ANM_AnimationLoaderApplet.TargetFrame=_top
AppletViewerPopup.ANM_AnimationLoaderApplet.BorderWidth=2
AppletViewerPopup.ANM_AnimationLoaderApplet.BorderType=Standard
# C. Ad Applet MediaList
AppletViewerPopup.ANM_AnimationLoaderApplet.mediaURLList=new
AppletViewerPopup.ANM_AnimationLoaderApplet.mediaURLList.size=2
AppletViewerPopup.ANM_AnimationLoaderApplet.mediaURLList.element0=deepsea.anm
AppletViewerPopup.ANM_AnimationLoaderApplet.mediaURLList.element0=animApplet.jar
```

2000

POINTCAST JAVA広告のAD DESCRIPTORファイルコンテンツの例1

フロントページの続き

(74)代理人 100096781
弁理士 堀井 豊
(74)代理人 100096792
弁理士 森下 八郎
(72)発明者 ランズマン，リック・ダブリュ
アメリカ合衆国、１０５９７ ニュー・ヨーク州、ワッカブク、ラムパート・パス、８
(72)発明者 リー，ウェイーイエー
アメリカ合衆国、１００１９ ニュー・ヨーク州、ニュー・ヨーク、ダブリュ・フィフティエイス・ストリート、１４０、アパートメント・９－シィ

審査官 金子 幸一

(56)参考文献 国際公開第９７／０１５０２０（WO，Ａ１）
特開平１０－２５４９６６（ＪＰ，Ａ）
特開平１０－２００４３０（ＪＰ，Ａ）
特開平１０－０９１５８０（ＪＰ，Ａ）
特開平０９－１３５２６５（ＪＰ，Ａ）
特開平０９－０９１２１５（ＪＰ，Ａ）
特開平１０－１３３８３８（ＪＰ，Ａ）
特表平１１－５１３８２９（ＪＰ，Ａ）
特表２００１－５２５９５１（ＪＰ，Ａ）
Andrew M. Fedorchek ほか，Professional Library ASP，１９９８年 ４月２０日，p.152-171
佐藤 ほか，WWWにおける映像データへのリンク情報付与のための枠組み，レクチャーノート／ソフトウェア学１８，近代科学社，１９９７年１２月２０日，p.23-30

(58)調査した分野(Int.Cl.，ＤＢ名)
G06Q 30/00
G06Q 10/00