September 22, 2017

Verifying Client Media Playback using Unique Embedded Metadata

Benjamin Seligman

Christopher Dinn

Damien Levin

Verifying Client Media Playback using Unique Embedded Metadata

ABSTRACT

The described subject matter is directed to injecting metadata into a media stream at (or just after) a cache edge of a content delivery network ("CDN"). A unique instance value (e.g., included within a tag or uniform resource locator ("URL")), is embedded by an edge device into metadata for the media stream. Examples of media stream metadata include, but are not limited to, D3 timed metadata inserted in an MPEG-2 transport stream or an emsg box in the ISO-BMFF video container for MPEG-DASH. When a streaming media player receives the metadata, it generates a network request or callback request including the tag or directed to the embedded URL. A server receiving this message can use the unique instance value as confirmation that the player received and processed the metadata, which indicates that media content delivered with the metadata was presented by the media player. The server can then authorize additional media delivery and/or generate reliable presentation statistics for the content.

DESCRIPTION

Streaming media, e.g., audio or video streams, may be delivered via a network, e.g., the Internet, from one or more media servers to client devices for presentation by media player components of the client devices. A media player receives the streaming media as a series of packets, blocks, chunks, or other units. The media player receives this series of data and uses the received data to construct the media for presentation. In some instances, the presentation may be effectively in "real-time," presenting the media as the data arrives (or shortly after the data arrives) at the client device. The media servers and the media players may communicate using a customized protocol or a standardized protocol such as Real-time Transport Protocol ("RTP") or Dynamic Adaptive Streaming over HyperText Transport Protocol ("DASH," also known as "MPEG-DASH").

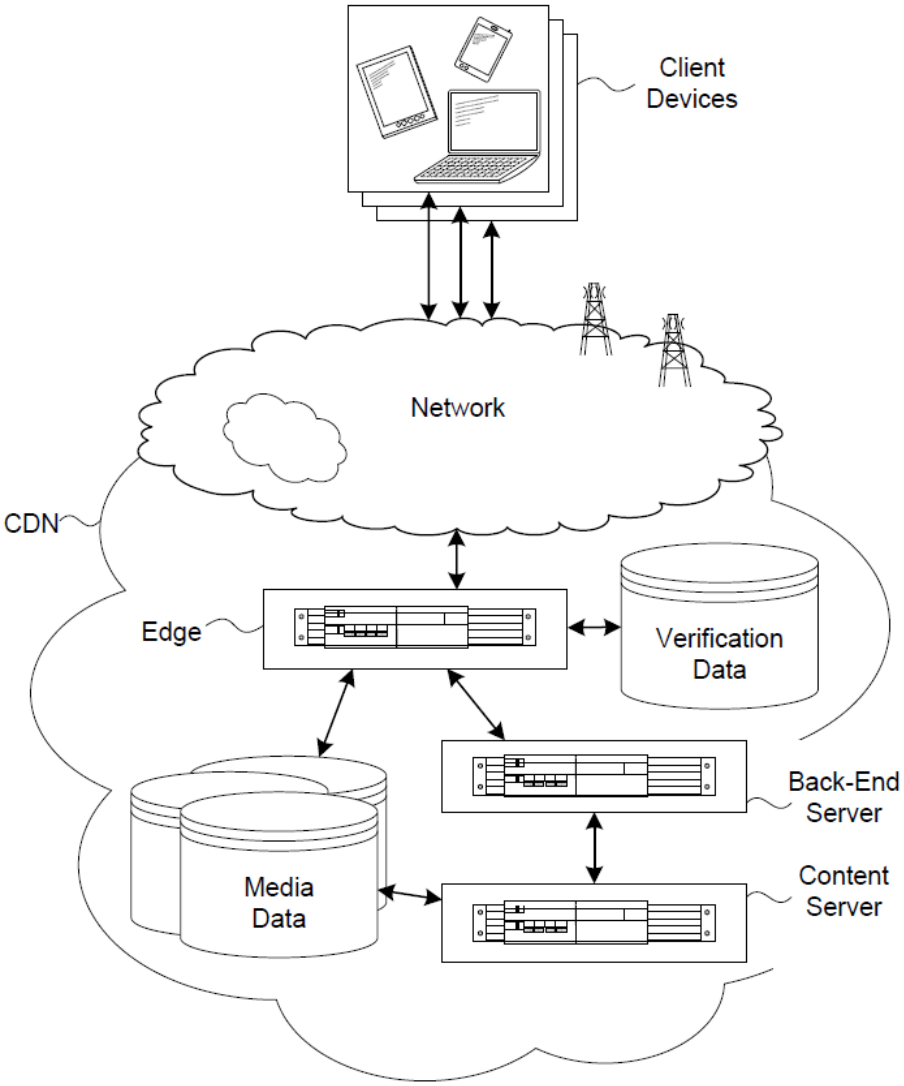
In some instances, the media player may be configured to generate a sequence of requests for the media stream. For example, a media server might first transmit, to the client device, a manifest including a list of identifiers for the media player to request. An identifier may be, for example, in the form of a uniform resource identifier ("URI") or locator ("URL"), with network

address information, protocol information, and embedded identifiers and parameters. Some of these identifiers may correspond to content that is added to the media stream during presentation. For example, the media stream may include a first content item such as a first portion of a video, a second content item such as additional content that may be selected separately for delivery with the video, and a third content item such as a second portion of the video for presentation after presentation of the additional content. However, usually, there is no guaranteed correlation between the media content downloaded by a player (e.g., chunks downloaded) and what is actually rendered by the player (e.g., chunks presented).

Verifying that the media player at the client device actually presented the various content items is difficult because the media player is outside the control of the media servers. In some instances, a media player may be configured to skip or fast-forward through some portions of a media stream. In some instances, a media player may be configured to rewind or replay a portion of a media stream. These non-linear presentation events can complicate presentation timelines and make it difficult to know if media content delivered to the client device was presented, and if so, how it was presented. However, it may be important to verify that the client device received each content item and that the media player presented each content item. For example, there may be license fees that must be paid based on presentation of the second content item, or placement fees to be collected based on presentation of the additional content. Known solutions provide insufficient verification or interfere with data distribution and delivery mechanisms. For example, media servers may serve the same source media data to a variety of client devices. Accordingly, the source media is not specific to the recipient client devices. Generalized source data may be cached along the distribution channel, but this mechanism fails if the data is not generalized.

Accordingly, a technical verification solution is needed that does not interfere with the data distribution and delivery mechanisms while uniquely identifying events that occur during presentation of streaming media content.

The following figure depicts client devices in communication with a content delivery network (“CDN”) via a data network such as the Internet:



Above is a block diagram illustrating an example computing environment that includes a network, e.g., the Internet, through which client devices communicate with the other network-connected computer systems, such as the elements of the content delivery network ("CDN"). The illustrated CDN includes an application back-end server, a content server managing media data, and an edge device managing verification data. The elements of the CDN may be within a single autonomous system ("AS") network or distributed across multiple networks.

Suitable example client devices include a computing system, or processor-based device, that executes applications, presents output to a user, and receives input from the user. The client device may be any kind of computing device, including, for example, a desktop computer, a laptop or notepad computer, a mobile device such as a tablet or electronic pad, a personal digital

assistant, a “smart” phone, a video gaming device, a television or television auxiliary box (also known as a “set-top box”), a kiosk, or any other such device capable of exchanging information via the network. The client device includes one or more hardware elements for facilitating data input and data presentation, e.g., a keyboard, a display, a touch screen, a microphone, a speaker, and/or a haptic feedback devices. The client device may exchange information over the network using protocols in accordance with the Open Systems Interconnection (“OSI”) layers, e.g., using an OSI layer-4 transport protocol such as the User Datagram Protocol (“UDP”), the Transmission Control Protocol (“TCP”), or the Stream Control Transmission Protocol (“SCTP”), layered over an OSI layer-3 network protocol such as Internet Protocol (“IP”), e.g., IPv4 or IPv6. The client device may receive streaming media via a customized protocol or a standardized protocol such as Real-time Transport Protocol (“RTP”) or Dynamic Adaptive Streaming over HyperText Transport Protocol (“DASH,” also known as “MPEG-DASH”). The client device presents the streaming media via an application generally referred to as a media player. The media player may be a stand-alone application or may be integrated into another application. In some instances, the media player is a component or element of a browser application (e.g., a web browser) capable of receiving data formatted according to the suite of hypertext application protocols such as the Hypertext Transfer Protocol (“HTTP”) and/or HTTP encrypted by Transport Layer Security (“HTTPS”). In some such instances, the browser facilitates interaction with one or more servers via interfaces presented at the client device in the form of one or more web pages. In some instances, an application executed by the client device communicates with the CDN using a custom instruction set, e.g., defined by an application programming interface (“API”), where the application executed on the client device implements the API. An application can implement an API using, for example, a library or software development kit (“SDK”) provided to the application’s developer.

The application back-end server provides back-end support to an application executing on the client device. In some implementations, the application back-end server runs a service that receives data from the client device and sends data to the client device, or causes data to be sent to the client device. For example, the client may execute a media-player that interacts with the application back-end server to request media content, which is then delivered by the content server, e.g., from the media data storage. In some implementations, the application back-end

server receives content requests, authenticates access credentials (if any), validates content requests, and regulates or manages how a content server provides content responsive to a request. For example, in some implementations, there may be multiple content servers distributed geographically such that some content servers are more proximate to a particular client device or otherwise better situated to service a particular client device (e.g., having less latency or greater bandwidth between the respective content server and the particular client device.) The application back-end server identifies a content best situated to service a particular request and assigns the identified content server to service the request. In some implementations, the application back-end server directly instructs the identified content server to service a particular request. In some implementations, the application back-end server directs a client device to submit a request to the identified content server.

In some versions, the application back-end server selects media content for delivery to the client device. For example, the client device may submit a request for a media program and the application back-end server may respond by identifying specific media content for the media program and, in some instances, additional media content to provide in addition to the requested media program. The application back-end server may determine to insert the additional content in between portions of the media program, to provide the additional content prior to the media program, to provide the additional content after providing the media program, or some combination thereof. In some implementations, a separate content selection server (not shown) determines what additional content to provide. For example, additional content may be selected based on parameters taking into account the client device, the media player, context of the client device or media player, time of day, geography, the requested media program, and so forth. In some implementations, an automated auction is used to select the additional content.

The content server, like the application back-end server, provides server-side functionality to an application executing on the client device. In particular, the content server transmits media data from the media data storage to the client device via intermediary network devices such as the illustrated edge device. In some versions, an application executing on the client device generates and transmits a request to the application back-end server, which then instructs a content server to service the request. In some versions, an application executing on the client device generates and transmits a request to the application back-end server, which then

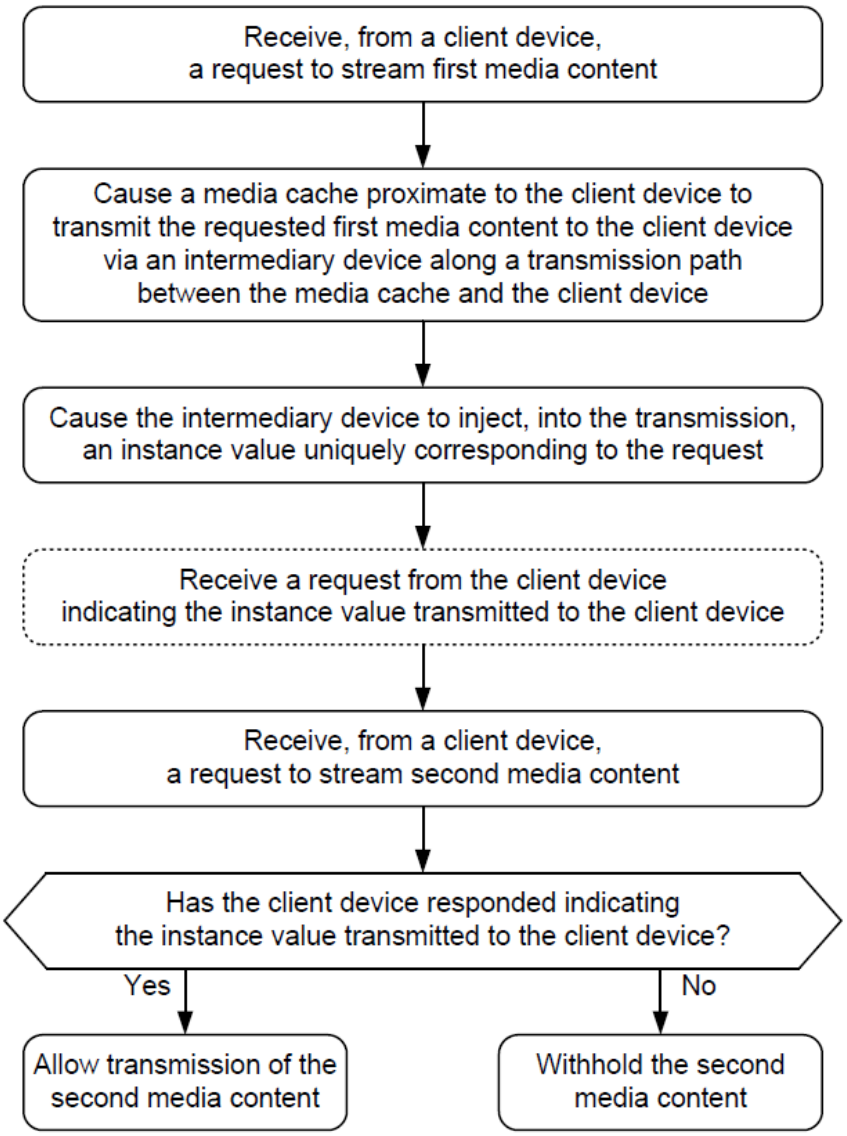
instructs the client to resolve the request using a particular content server to service the request. The content server may be part of a distributed media cache. The application back-end server and the content server may be the same server, or are operated or managed by the same party, e.g., in the same autonomous system ("AS") network. The application back-end server and the content server may be distinct servers and, in some implementations, may be operated or managed by different parties. For example, the content server may be controlled by a third-party providing media caching services.

The content server transmits media content from the media data storage to the client devices as a stream, e.g., using a customized protocol or using a standardized protocol such as Real-time Transport Protocol ("RTP") or Dynamic Adaptive Streaming over HyperText Transport Protocol ("DASH," also known as "MPEG-DASH"). The stream is transmitted through an intermediary network device (e.g., the illustrated edge device) that receives data (packets, chunks, blocks, etc.) for the streaming data transmitted by the content server to the client device and may modify the streaming data prior to forwarding the streaming data to the client device. For example, the streaming data may include metadata (or include data fields for metadata) and the edge device may modify or augment the metadata. The edge device inserts a value into the metadata that is unique across all streams from the CDN, that is, the value is unique to the particular instance of the stream. The unique instance value corresponds to the particular instance of streaming media. In some versions, the unique instance value is further unique to the particular block or chunk data into which it is injected.

The unique instance values inserted or injected by the edge device may be retrieved from, or stored in, the verification data. In some versions, the value is a randomly generated number. In some versions, the value is a more meaningful or specific value, such as a data tuple including, for example, a delivery timestamp, a content identifier, and a destination identifier. In some such versions, the data tuple is encrypted or hashed, such that the internal values of the tuple are secured. In some versions, the unique instance value is stored in association with data describing the streaming instance. For example, the value may be stored in association with an identifier for the media stream program, with an identifier for the media player (e.g., a generalized identifier that describes the player itself and/or a more specific identifier tied to a particular instance of the player), with an identifier for the client device, with an identifier to an

account associated with the client device and/or with the instance of the media player (e.g., a media access account), with time and date data, with geo-location data, with network data, and so forth. Because this value is injected into the stream by the edge device, at the edge of the CDN, the media presented may be sourced by the CDN from a generalized cache, e.g., from media data stored within the network at a location physically, geographically, or technically close or proximate to the client device. For example, the cache may be considered proximate if the latency between the cache and the client device is lower than other available caches storing the same media content.

The unique instance value injected into the media stream may then be used to determine how the media stream is presented at the client device. In particular, the value may be embedded in data that the media player, at the client device, will transmit back to the CDN when corresponding media is presented. For example, in some protocols, a chunk of media data may include a tag or other metadata that is processed when the chunk is rendered and presented. Upon processing, the client device may issue a request to a uniform resource locator ("URL") included in the metadata, or may otherwise generate a data request to the CDN based on the metadata. Accordingly, the unique instance value may be returned to the CDN when the metadata is processed, which only occurs when the media chunk hosting the metadata is rendered and presented. In some instances, the reporting structure (e.g., the URL) reports the unique instance value and a second identifier corresponding to the portion of the media rendered. Because the value is unique to the particular stream, and in some cases unique to a particular portion of the stream, the CDN can then determine when the portion of the stream is presented at the specific client device corresponding to the unique value. The following flowchart illustrates an example of how this may be used:



In the flowchart above, a CDN provides a media stream to a client device based on receipt of the unique instance value. As shown, a CDN receives, from a client device, a request to stream first media content and, in response, causes a media cache proximate to the client device to transmit the requested first media content to the client device via an intermediary device (e.g., the edge device) along a transmission path between the media cache and the client device. The CDN causes the intermediary device to inject, into the transmission, an instance value uniquely corresponding to the request. The CDN should subsequently receive a message from the client device indicating the instance value transmitted to the client device, when the client device renders the media. The CDN receives, from the client device, a second request to

stream second media content and determines whether the client device has responded indicating the instance value previously transmitted. If so, then the CDN allows transmission of the second media content, e.g., in the same manner as before, and otherwise the CDN withholds the second media content. Thus delivery of the second media content is contingent on rendering and presentation of the previously delivered content.

In one example, the CDN delivers a manifest to the client device responsive to a request to stream a media program. The manifest includes identifiers for portions or chunks of the media program and identifiers for additional content to be delivered with the requested program. The media player executed by the client device then generates requests for each portion identified in the manifest. The identifiers in the manifest may be generalized, e.g., referring to cached portions of the media program such that one source copy may be used to service multiple clients. The CDN receives a request for a first portion and, as shown in the flowchart, causes a media cache proximate to the client device to transmit the requested content to the client device via the edge device. The edge device injects a unique instance value into the streamed content, e.g., at the cache edge. As described above, the edge device may inject the unique instance value by modifying metadata in the transmitted data after it leaves the cache but before it is delivered to the client.

The client device receives the modified content data and only processes the unique instance value when the media player presents the content data. Accordingly, the CDN only receives a message or indicator from the client device including the unique instance value when the media player presents the associated content. The edge device stores the unique instance value (or uses one already stored) in the verification data. The CDN, upon receiving the confirmation (i.e., the request, message, or other indicator) from the client device that includes or indicates the unique instance value, may store a message or event describing receipt of the confirmation. The CDN may then use the confirmation to authorize additional content delivery, to increase a license fee payment owed to a content source, and/or increase a charge due for paid content placement. In some instances, the CDN may receive the confirmation unexpectedly, e.g., some significant amount of time after delivery of the unique instance value. This may indicate fraud, copyright infringement, or problems with the CDN or network infrastructure. In some such instances, some versions may generate an alert or block further delivery of content.

In some instances, the CDN generates statistics data from the verification data. The statistics data may include stats based on the content transmitted to client devices and stats based on the confirmation messages received. For example, the stats may include differentials for delays between when a unique instance value is transmitted to a client device and when a callback confirmation message or indicator is received from the client device with the unique instance value. In some instances, the stats may include data regarding unique instance values delivered to client devices but never returned. The various stats data may then be included in reports or dunning statements.

One advantage of the described approach is that media players do not need to include complex verification schemes. The media player simply returns the tag or URL that includes the unique instance value. This means that third-party application developers do not need to incorporate black-box code or link in developer tools that might force alternate development cycles. The verification is server-side, simplifying application development. Another advantage is that verification can be client-specific while allowing content delivery from a cache-based system. A CDN may use multiple media caches distributed throughout a network. Edge devices between the caches and the client devices handle the verification.