Recurring License Renewals for Concurrency Detection and Enforcement

Introduction

Streaming Over-The-Top video delivery services often need to restrict the number of concurrent playbacks per user account. This document describes the use of license renewals (aka “heartbeats” aka recurring license updates) as a mechanism for detecting and enforcing such streaming concurrency limits.

Recurring License Renewals is a feature implemented server-side and supports both continuous detection and enforcement. This solution provides the following benefits:

- A great user experience as it does not impact well-behaved users (most users using OTT video delivery services) and allows placing restrictions on users displaying suspicious behaviors quickly.
- Scales effectively across types and number of clients as it is independent of client architecture, does not require any type of storage in clients and could be made to work effectively in private browsing modes and on stateless devices.
- Provides flexibility for authors to architect their service as threats evolve.
- Effectively addresses detection and enforcement with a single solution.
- Is consistent with the principles of the Web Platform.

Below, we provide details on how recurring license renewal works, describe the flexibility content providers have in implementing policies, explain how server outages can be handled along with an example flow, and highlight benefits of this solution, especially for scalable internet-based video on demand services.

How Recurring License Renewal Works

When a media license is requested by a DRM client, the license server responds with a limited-duration license. The license usually has a duration on the order of a few minutes and is configured with a policy that allows for license renewals. Well before the license expires, the DRM client issues a license renewal request (aka heartbeat request), which may be repeated if a valid renewal is not received after some configured timeout. When the DRM server services the license renewal request, it may choose to extend the license for another license renewal period or to discontinue key usage rights - by means of a zero-duration license or similar. When the DRM client receives the license renewal, it extends the license by the prescribed amount, or revokes access to the keys, thus terminating playback.

Detecting concurrency

The DRM servers can easily determine how many concurrent devices are accessing a particular piece of content - or the service in general - by counting the renewals requests that were received in the last renewal period for a user and/or piece of content. If the DRM servers have not heard from a DRM client instance within the last renewal period, they may assume that that device is no longer playing back the content because either the user has discontinued playback or its key usage rights have been revoked due to license expiration.

The implementation on the servers can be rather trivial. The servers servicing license renewals can keep a queue of recorded heartbeats in which old entries expire after a time equal to the license duration. The main DRM server can decide whether to honor a new initial license request by checking the license-renewal server queue to determine concurrent ongoing playback sessions. Alternatively the main DRM servers can always service new requests, while instructing the renewal servers to discontinue license rights for the oldest session next time a license renewal request is received for said session. This has the advantage of always granting the most recent user request (vs. denying a new request because other playbacks are believed to be in progress). Alternatively, a detection-only mode can be implemented by having both the initial license request and renewal servers always service license requests and just logging the request.

Adaptability

The license period can be adjusted dynamically and per account/device/title based on any parameters the content provider chooses. Such adjustments might include increased periods during times of server overload, data center outage, etc.; increased periods for “well-behaved” accounts; decreased periods, different policies, and/or additional scrutiny logic for “suspicious” accounts or behavior; and optimistically granting a license or renewal in abnormal situations for “well-behaved” accounts.

Both the renewal request interval and the license duration are configurable. These intervals can be varied to achieve various goals.

The following example illustrates a model providing three levels of enforcement, with all users starting in detection-only mode:

1. Detection-only mode:
    - renewal interval = 5 minutes, license duration = content duration + x minutes (for pausing, etc)
    - These settings offer no reduction in reliability vs. other mechanisms, such as secure release, due to license server outages.
    - If the account generates more than a certain number of new license requests (across multiple devices) and no renewals in specific period of time, enable lightweight enforcement due to possible suppression of renewal requests.

2. Lightweight enforcement mode:
    - renewal interval = 5 minutes, license duration = content duration / 2
    - If the user is really suppressing renewal requests, it now becomes impossible to watch any piece of content in one shot.
    - If the account consistently generates two or more new license requests for the same content on the same device, or there is a suspicious renewal request (time since last renewal is close to the license duration instead of 5 minutes) move the account to strict enforcement, as this suggests the user is suppressing renewal requests and simply reloading halfway through the content (when the license expires).
    - Accounts in this state are only slightly more impacted by server outages as only an extended outage halts playback.
3. Strict enforcement mode:
    - renewal interval = 3 minutes, license period = 5 minutes.
    - The user cannot watch more than 5 minutes of content at a time if suppressing renewal requests.
    - In this mode the user may be affected by server outages of more than 2 minutes.
    - After some period of time (a week, etc), the user may be dropped down to more relaxed modes.

Regardless of the heuristics used, the user experience for accounts under more relaxed enforcement is not impacted, and the impact on server load is negligible because most users fall in this category. For non-abusing users, being in a more restrictive mode only negatively affects the user if there is a major license server outage.

Resilient, Independent License Servers

While a simple implementation of license renewals may require that servers and clients have shared knowledge of session keys used to authenticate messages, it is possible to instead compute such keys such that both the license server servicing an initial license request and the server servicing a license renewal request can derive the same session keys and authenticate and sign messages without having to exchange any information between them. Thus, even if a license server is unable to access the infrastructure required to service an initial license request, it can still service license renewal requests. In addition, the license servers can be designed to always renew licenses in an emergency, such as when the backend infrastructure becomes unavailable, rendering them unable to check concurrency.

In such an implementation, as long as the license servers are running and reachable, users' existing playbacks will not be interrupted. Resiliency can be further increased by adding license server redundancy and including multiple URLs in the application for handling license renewal messages.

Reliability can be further increased by deploying renewal servers across geographies and infrastructures. This maximizes the chances that license renewal requests will be serviced should central servers or the original data center become unreachable.

License renewals are usually “lighter” than initial licenses, as they only contain some policy data indicating to extend or revoke usage of the media keys, and not the actual media keys. This means that renewals require fewer resources to service than initial licenses. Renewal servers can also be detached from the key server/store as well as, for example, the main user account database. Therefore the license renewal servers may be different from the those servicing the initial license request.

When a server goes down...

The following is an example of how a content provider may choose to implement a very reliable concurrency-enforcing system using distributed redundancy. It is relevant mostly for users which are in strict enforcement modes, as users in detection-only or lightweight enforcement modes are not adversely affected by server outages. Other designs are also possible, including less complex designs or ones that only detect concurrency.

When the license server servicing an initial license request becomes unavailable:

- Initial license requests from the application will fail (i.e. 503 or some other error).
- Seeing a network failure for a “license-request” message, the application requests the license using the URL for a fallback license server.
    - The application can have any number of fallback URLs, including ones for different clusters or a data center across the country or in another country or continent.
- The application keeps trying additional fallback URLs until one succeeds.

When the license server servicing a renewal becomes unavailable:

- Renewal requests from the application will fail (i.e. 503 or some other error).
- Playback will continue for the duration of the license, which is likely a few more minutes.
- Seeing the network failure for a `"license-renewal"` message, the application requests a renewal using the URL for a fallback license server.
    - The application can have any number of fallback URLs, including ones for different clusters or a data center across the country or in another country or continent.
- The application keeps trying additional fallback URLs until one succeeds.
    - If any URL is reachable, the application has more than enough time to attempt to connect to a number of URLs before the license expires.

The license servers employ the following logic when servicing initial license requests:

- The license server accesses the user database and content key store to fulfill the license request.

    - If the license server backend is unavailable, then the server rejects the license request (i.e. 500, 503, or some other error).
- The license server checks the record of heartbeats or concurrent usage.
    - If the concurrency infrastructure is not available, the server may choose to service the request, or fail (i.e. 500, 503, or some other error).
- Based on the concurrency information, the license server may reject or service the license request. Alternatively it may always service new license requests, while flagging an older license for discontinuation.

The license servers employ the following logic when servicing license renewal requests. Note: Any license server can service any renewal request.

- The license server checks some record of heartbeats or concurrent usage.
    - If that record is inaccessible for some reason (i.e. that server is down), the server determines whether it should enter an emergency renewal mode.
        - For example, repeated attempts to reach a backend component fail or some other “circuit breaker” is tripped.
        - Since this is a rare event, such detection would be logged, trigger alerts, etc.
    - If in emergency renewal mode, the renewal server will renew all valid licenses.
- If the session is flagged for discontinuation, the server denies the license renewal request.
- If desired (for example in emergency renewal mode or due to server overload), the renewal specifies an increased license duration.

Benefits of License Renewal

The following are some of the benefits of of license renewal.

- Scales effectively across any number and type of clients.
    - Can be supported in any implementation that supports the simple streaming case. That is, all user agent implementations that support EME.
    - Does not constrain client implementation or device architecture. If a client can request and handle licenses, it can support license renewals!
    - Does not require any type of persistent storage, even JS storage, on the client.
    - Can work in private browsing modes (assuming other privacy issues are addressed by the implementation) and on stateless devices.
- The lack of a renewal request before expiration is the “proof” that the content is no longer being played. Unlike, for example, secure release, there is no chance of lost proofs due to laptop lid closure, crashes, etc.
- License renewals place the complexity of the solution on the DRM servers, where the complexity (little as it is) needs to be addressed just once.
- Does not require tracking session history across days or weeks to correlate delayed or lost release messages.

- Flexibility for authors: As with the rest of the web platform, a server-based solution can be updated, tweaked, experimented with, and adapted to emerging use cases or threats without having to update every client implementation.
- Simplicity for authors: A single solution for enforcement of concurrent license limits.
    - Use of secure release still requires such a solution for enforcement for suspicious users.
    - License renewals is a single solution that can be tweaked per account, if desired, and provide direct feedback about the impact of those tweaks.
    - Avoids the risks of having a second rarely-used path or switching users from one to the other.
- Provides near real-time usage metrics and enforcement.
- Requires no explicit changes to or text in the EME spec, though we don't oppose explicitly covering it in the spec if desired by the group.