Http caching - Mechanism improvement for managing content expire headers

The concept could be looked to be similar to cache manifest file, but with more flexibility. An external content upto-datefile contains regular expression, which identify resources under its current url location.
For each regular expression, content upto-date(content expiry) details can be specified or abstracted using a label system. External content upto-date files, should be able to include other external content upto-date files, as if they were inlined from the start. Content upto-date details can be any of the traditional existing content/cache expire headers with the additional of modstamps, which is similar to E-Tag, which allow server to push updates.

When the server wants to force evict or change a cache item the a new upto-datefile can be pushed from the server, obviously.

Load Balancing incorporation

Updating the upto-datefile could cause many files and resource to be requested from the servers, putting them under extremely high loads all at the same time. Imagine a simple modstamp that is abstracted by a label, being changed, the knock on effect can be hugh. As a single server could server this simple update to millions of clients per second, but not be able to deal with the load it creates on their system.

How can one address this:

For each entry a label could be specified, which is used to group items by some common CDN, Amazon availability center, server data center type to serve the different request types. For each load group one now needs to communicate some predictable method of orchestrating the request rate.
If one makes the assumption, that a client needs all of the resource that were invalidate to be download to complete, one would think you want to

rate limit per client. One thing about rate limit per client is that each client's bandwidth is different and can rapidly change, especially on a mobile.

Since can’t really control the load on the services remotely, the only thing I can think of is to introduce a upto-date manager. Request would be made to the uptodate manager which would determine the load for each load group.
The manager, would then control the rate at which the uptodate resource should be made available to clients, my monitoring the current load, to ensure the servers are not overloaded in any of the working groups.

This is not entirely a fail safe, as some resource mayn’t be requested immediately, but could where alot of common resources.

The other thing to look at is the ability to dynamically vary the content expire headers time values, were could have priority of resource and acceptable ranges, in which attempt to load balance, by building a histogram of the request of the resources over x minutes, and then attempt to pick expire time which averages the history gram, such when content expires and the request hits the server again, it's out of phase causing things to load balance. Extending the context expire headers timer with out look at previous load and balancing the history gram, will just result in a phase shift of the load.