Technical Disclosure Commons

Defensive Publications Series

December 13, 2016

Synchronized Chat Notifications

Christopher Wren

SYNCHRONIZED CHAT NOTIFICATIONS

ABSTRACT

This disclosure presents a system and a method to generate or control synchronized notifications in co-located devices. The method is implemented via suitable software such as an app. The method assumes that the devices have clocks that are synchronized (at least at a human-perceptual-scale), and are therefore likely on the same network. In this case, the server assigns a date certain to the notification data received and the app simply waits until that time to execute the notification. The method ensures that the notification sounds will all emit at very similar times. A device that experiences network delays would be disregarded by the app and will not be part of the synchronized notification.

BACKGROUND

Chat groups often contain people who are also (at least partially) co-located physically: family chat groups, work-group chats, etc. When a message is sent, the various attendees' devices post notifications independently, usually in response to a network sync event. This results in a notification storm with slight delays between each alert in any physical location where there are multiple participants present.

Typically, a notification goes through the following steps:

1. server receives an update and posts a group chat message in response.
2. app receives the group chat message update that there is new data.
3. app performs a network request to get the data.
4. app receives and parses the data.
5. app formats one or more notifications corresponding to the data.
6. app posts the notifications to the system.
7. system determines if the notification should generate an alert tone.
8. system generates the alert tone and displays the notification.

Small messages may be included in the group chat message itself, collapsing steps 2 through 4 above. The largest sources of delay and variability come from steps 1-2, 3-4, and 7-8, with the first two being the most problematic by far. If the time for one or more devices is larger than a few seconds, the multiple near-simultaneous notifications could mask useful information such as a reply. The multiple notification alert tones could also be quite annoying.

DESCRIPTION

This disclosure presents a system and a method to generate or control synchronized notifications in co-located devices. The method is implemented via suitable software such as an app, and considers two cases: A) the time between steps 1 and 8 for all devices involved is small: a second or two at most (and is often less), and B) the time for one or more devices is larger than a few seconds. The method envisages that the app has control of the processes up until the notification step (Step 6 above), so the app can independently mediate any volatility before that step.

The method assumes that the devices have clocks that are synchronized. This is reasonable because the devices are co-located and are therefore likely on the same network (office or home wifi for example) and therefore have nearly identical access to NTP servers and the like. In this case, the server assigns a date certain to the notification data received in step 4 and the app simply waits until that time to execute step 6. Assuming that the delay from 6-8 has little to no variability, the method ensures that the notification sounds will all emit at very similar times (Case A).

When the time for one or more devices is larger than a few seconds (Case B), delaying the notification of a chat message by more than a couple of seconds will annoy users, as it violates the “instant” part of the contract implied by “instant messaging”. If one or more devices are lagging behind by more than a few seconds due to network delays, the method disregards them and only processes the rest of the notifications as for Case A. The relevant user would know that their device is having trouble and a late notification from such device will not be perceived as part of the annoying notification storm. Any device that receives the update in step 4 after the date certain is considered to have missed the deadline due to network delays and is not included in the synchronization. The device just posts the notification as soon as possible.

The system and method can compare the inter-arrival times of the requests from step 3 onwards between devices and use that to determine how much time to allow for the prescribed delay. The clients could even include information in the request (3) about how much spare time they had on previous notifications (or if they missed deadlines), network stats, click skew stats, etc. In another variant, the method or application could use co-location to determine which clients are in close proximity, allowing the server to partition them into groups for the purpose of scheduling notifications. This will ensure that a lone device doesn't have to delay at all, for example. In another case, a group of devices don't have to wait to accommodate a device that is in another location (possibly on a slow or marginal mobile network, for example).

In a variant of the method, the notifying apps could use co-location to schedule their own notification time. They would discover each other and then compare clock skews and other sources of delay similar to synchronization of network broadcasts. The system and method disclosed could provide support for a "post at time" api in step #6. That would allow the app to eliminate variability in timing of steps 6-8. This is currently a small source of variability, but it may be necessary to implement this as mobile operating systems become more aggressive about scheduling app processes to save battery, thereby injecting variability into the app-controlled delay between the formatting (5) and notification (6) steps.