Posted in: Development (/spaces/127/sandpiper/forums/5366/development)

Polling vs Message Queues

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)
12/22/2019

The client process controls the sync process. It's responsible for deciding which data-objects are new and which ones need to be deleted from their synced data pool. The server simply responds to client requests about what is stored in its (primary) data pool.

But how does the client know when to start a sync process? Ideally, it would **initiate a sync only when there's been a change**. For this to work, the server needs to notify interested clients that something has changed. This "messaging" approach would also satisfy the Sandpiper "real-time" delivery goal.

As an alternative to "real-time" delivery, we could design the client to periodically ask the server if anything has changed. (Or, more correctly, it would periodically ask the server for information and decide for itself if something had changed.) This periodic process is called "polling".

There are usually two types of message delivery approaches:

1. **"at-most-once" delivery.** If a subscriber is unavailable, it will not receive messages.
2. **"at-least-once" delivery** (aka Guaranteed Delivery). Messages are persisted until successfully delivered (or until resource or defined limits are reached).

The second method is generally more difficult to implement and deploy.

Our plan is to support both polling and "at-most-once" (message-based) sync initiation. We're looking at NATS (https://nats.io/) for server notifications to subscribed clients. NATS is open-source, cross-platform and supports "at-least-once" delivery (as a future Sandpiper enhancement).

We are also using websockets for the sync process itself, so we might consider that transport as a messaging system for issuing an "at-most-once" event to initiate a sync.

messaging (/spaces/127/sandpiper/searchresults?keyword=%22messaging%22&searchtags=1)
nats (/spaces/127/sandpiper/searchresults?keyword=%22nats%22&searchtags=1)
polling (/spaces/127/sandpiper/searchresults?keyword=%22polling%22&searchtags=1)
real-time (/spaces/127/sandpiper/searchresults?keyword=%22real-time%22&searchtags=1)
sync (/spaces/127/sandpiper/searchresults?keyword=%22sync%22&searchtags=1)
websockets (/spaces/127/sandpiper/searchresults?keyword=%22websockets%22&searchtags=1)

Like

Reply (/forums/post?tid=5413&ReplyPostID=5414&SpaceID=127)

Luke Smith (https://autocare.communifire.com/people/autopartsource)
12/23/2019

This is why I was so wound-up about client vs server roles in the beginning. If the publisher is represented by a "client" (in the TCP client/server sense), then the author's changes can literally be pushed in real-time. This is because clients are responsible for initiating socket connections - not servers. Servers quietly wait forever listening for clients to connect. We've said that the publisher is always represented by a server - therefore we have committed to either frequent polling by receiver clients or some type of out-of-band messaging channel.

I think that message queues will add too much complexity and third-party package dependency. Polling (initiated by the client) can be done at a high enough rate that it approximates real-time for our purposes. In my world of brake pads, a daily poll/sync would be "real time". We're not talking about logistics data or payment processing.

Like

Reply (/forums/post?tid=5413&ReplyPostID=5415&SpaceID=127) Answer

Doug Winsby (https://autocare.communifire.com/people/dougwinsby)

12/23/2019

I agree that daily polling for "Level 1" (file-based) syncs should be sufficient.

One implementation question I have is whether or not the client should sleep between polls or poll then exit (to be started again by a cron job). I kind of like the idea of it always running, but we'll see what makes the most sense when we get to that point.

Like

Reply (/forums/post?tid=5413&ReplyPostID=5416&SpaceID=127) Answer

Krister Kittelson (https://autocare.communifire.com/people/krister-kittelson)
1/6/2020

I'm sorry to come late to this discussion; I was so focused on the 0.8 doc that I missed it.

One of the things that the plan establishes is a subscription schedule -- the polling frequency should match that. A snapshot holder can also request delivery by the server on a set schedule.

I was thinking of the sync schedule being definable in either seconds or days elapsed.

Like

Reply (/forums/post?tid=5413&ReplyPostID=5460&SpaceID=127) Answer