Push Notifications Compressor

1. Problem Definition

Some mobile apps like komoot provide the user with push notifications of their friends' updates like status posted … etc.

Although push notifications are useful, sending to many notifications can be annoying.

2. Proposed Solutions

A- Solution Idea:

To avoid such scenario in which the user receives a lot of annoying notifications, we came up with the idea of bundling notifications to reduce the amount of notifications we send.
That means we need to wait a bit to collect notifications until we can send those bundles. But we also know how important it is to send notifications as soon as possible; users want to know about friend's updates as soon as possible and start talking about it.
The goal is

B- Solution Requirements/Restrictions:

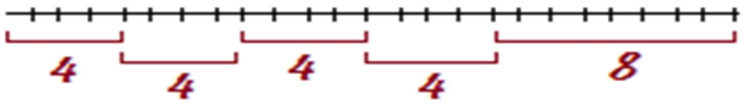
- To send no more than 4 notifications a day to a user (should happen only few times)
- To keep the sending delay minimal as possible.

C- Solution Criteria (what is guaranteed):

- The user will get all the notifications that happened today, today.
- The user will not get more than 5 push notifications per day.
- The average maximum delay between the update release (a user posted something), and receiving a push notification is 4 hours.
- The delay between the update release and receiving push notification with the updates will not be more than 8 hours ever.

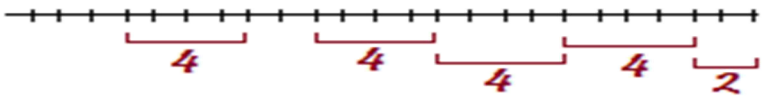
D- The Idea Behind:

If we splitted the day into 5 sections.



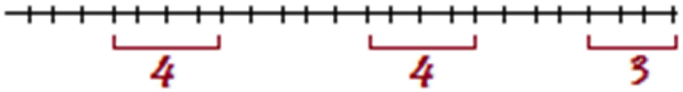
- The first section starts at 00:00, ends at 04:00.
- The second section starts at 04:00, ends at 08:00.
- The third section starts at 08:00, ends at 12:00.
- The fourth section starts at 12:00, ends at 16:00.
- The fifth section starts at 16:00, ends at 00:00.

This idea seems good, but a better modified version is, instead of putting specific milestones (time stamps) at 04:00, 08:00, 12:00 … etc, let's start when the first notification comes, so the distribution can be something like this (in reality):



Here you start counting the four hours limit (for each of the first four periods) just when you receive a notification, the same with the last period too.

Even if you received less than 5 notifications:



You still can take 4 hours (or less) starting from the first new notification.

E- The Implemented Solution:

Here I used a day based simulator, simulator that assumes that I cannot see the future notification (which is the actual scenario in real life), but I assumed that I can send the notification at any time just by specifying it (even after the time elapsed), but with no more than half a day maybe.

F- Idea Strength:

The idea's strength is that, it's dynamic, but once you received a notification, it will stay in the stack maximum 4 hours.

It can be better if we made some analytics over all the user experience, so we can know which period is better to send push notifications.

G- Idea Weaknesses:

At worst case, it will be five push notifications per day, and four hours difference between releasing a notification and receiving it – eight hours for the fifth notification; on the other side, this is not so bad and will happen very rarely.

3. Toolkit, Setup:

A- Toolkit:

Here I'm using:
- Python 3.5
- Pandas data frames library

Python; because it's easy to develop prototype with it.
Pandas; because it's so effective in processing data frames.

In reality, this problem may be better to be solved in Node.js because of the runtime streaming/processing, or C++ because it's faster in processing.

B- Setup the environment:

- You can find the code repo here:
https://github.com/AbdelrahmanRadwan/Push-Notifications-compressor

- Or download it as zip file from dropbox here:
https://www.dropbox.com/sh/6flgdg3secjxyob/AAAgewGZTA-mQsfnRq8bwBMxa?dl=0

- Install the Python requirements:

```
$ pip3 install -r requirements.txt
```

C- Running the code:

- Run the algorithm (it will ask you to choose the notifications csv file and will save the result to push_notifications_results folder):

```
$ python3 algorithms/push_notification_compressor.py
```

- Run the unit test (it will ask you to choose the push notification csv file and will save the result to tester/testing_results folder):

```
$ python3 tester/test.py
```

4. Code and Functionalities:

A- The push notifications generator:

In "\algorithms\push_notification_compressor.py", you can find the algorithm, let's discuss it simply:

A class user is created here, to hold and handle the user state regarding the bundled notifications, the received number of push notifications, friends who sent notifications and the earliest notification in the stack right now.

The user class contains these functions:
- Fire (): the responsible for sending the push notification.
- notify (): the responsible for managing the push notifications numbers and notification range (the 4-hour rule that we discussed before).
- new_notification (): the responsible for adding one notification insert the current range stack.

The flow of the algorithm:

- We go through all the records in the notification sheet (please make sure that the headers format like the following).

A	B	C	D
timestamp	user_id	friend_id	friend_name
8/1/2017 0:06	F62712701E7AF6588B69A44235A6FC	06D188F4064E0D47BD760EEFEB7AAD	Geir
8/1/2017 0:31	DF5BB50FAD220C8D2A8FF9A0DBAA47	588C89FCADD0DBA0E722822513A267	Antim
8/1/2017 0:35	8473CCCE79294CB494D1B42E2B1BAA	EDBB3D240ADBCF6CF175B192630ABB	Î£Ï‰ï„Î™Î¿Ï,
8/1/2017 1:20	CFFEC5978B0A4A05FA6DCEFB2C82CC	2BB0471CAA78ED0FCEE143E175F034	Mona
8/1/2017 1:21	0978C6F8C5093039165B5C571EACC8	45FE4C99C612BEEDE6A34B54C5369D	Laura
8/1/2017 1:21	FBA67EFA2766854B885F25C06CC2FA	92DEF3A48927B1B2B0295936679D1C	Rameshwor
8/1/2017 1:44	BE6B4CBB422BBF114FB109921F2B9F	7BCD287DF0EBF5CAA86458737777BD	NoÃ«
8/1/2017 2:09	391A4416FC0ADE8FD604B2F1A9BCCE	96593EE816FB4CE2AEBA5B754CFA38	Î›Îµï‰Î½Î¯Î±Ï,
8/1/2017 2:20	520EA69D3794B463EC553B79FDF850	723515D5D083C9C15EC9A24AA624D7	Lina
8/1/2017 2:20	A5C7951597609E2982344D66939AFF	723515D5D083C9C15EC9A24AA624D7	Lina
8/1/2017 2:20	C5F6178AF37CD86DF3B9D8CD811BAB	723515D5D083C9C15EC9A24AA624D7	Lina
8/1/2017 2:20	1B7EA1BAFAF72B4EA2AE1ACD8D775F	723515D5D083C9C15EC9A24AA624D7	Lina
8/1/2017 2:20	DF6A386FE701217C2A12292DB8D142	723515D5D083C9C15EC9A24AA624D7	Lina
8/1/2017 2:20	9BB953337ACF616EAEC93D46A45F20	723515D5D083C9C15EC9A24AA624D7	Lina
8/1/2017 2:20	349F293BA3435B7BC6AD35BF8396C6	723515D5D083C9C15EC9A24AA624D7	Lina
8/1/2017 2:20	C69E0349776A41A35F9929950CB2FD	723515D5D083C9C15EC9A24AA624D7	Lina
8/1/2017 2:20	651EA682AC26AA613FA8B143A4FB00	723515D5D083C9C15EC9A24AA624D7	Lina
8/1/2017 2:20	104CADE3535C93C8173036E682A9CD	723515D5D083C9C15EC9A24AA624D7	Lina
8/1/2017 2:20	08CD54CB5C552A6A4CBFFDD5D6B326	723515D5D083C9C15EC9A24AA624D7	Lina

- Every user of these users has unique ID, we map this unique ID to a user object (user defined class), to represent the user's state now(regarding receiving and aggregating the notifications).
- Firstly we check if this is a new day or not (in reality we will have a watch or clock running by default, so we don't need to do such check, I did it to avoid doing a long simulator), if it's a new day, then we have to fire all the push notifications hold from the previous day (this is mostly the last notification in the previous day), we fire it with a time period equals to the last minute in the preivious day or the time of the first notification in the stack now + 4 hours (in case that this is one of the first four notifications in the day and this is less than the end of the day).
- Now we notify the user, by calling the function notify in the user object, we check the time frame in which the user is experiencing now (based on the number of received notifications and push notifications in their state now), and decide if we have to fire the held notifications or to push the coming one to the stack (based on the time difference, is it more than four hours or not).
- If we reached the end of the file (this presents reaching the last few minutes in the day, in reality), we have to fire all the notifications held so far in users' stacks.

After running the script, it will ask you to provide the csv file which includes the notifications, and it will log to the terminal what is going on (the pushed push notifications), it also will save it to a csv file (/push_notifications_results/push_notifications.csv):

notification_sent	timestamp_first_tour	tours	receiver_id	message
8/1/2017 4:06	8/1/2017 0:06	2	F62712701E7AF6588B69A44235A6FC	Geir and 1 other went on a tour.
8/1/2017 4:35	8/1/2017 0:35	4	8473CCCE79294CB494D1B42E2B1BAA	Î£Ï‰ï„Î™Î¿Ï, and 3 other went on a tour.
8/1/2017 5:44	8/1/2017 1:44	1	BE6B4CBB422BBF114FB109921F2B9F	NoÃ« went on a tour.
8/1/2017 5:20	8/1/2017 1:20	6	CFFEC5978B0A4A05FA6DCEFB2C82CC	Mona and 4 other went on a tour.
8/1/2017 6:09	8/1/2017 2:09	2	391A4416FC0ADE8FD604B2F1A9BCCE	Î›Îµï‰Î½Î¯Î±Ï, and 1 other went on a tour.
8/1/2017 6:20	8/1/2017 2:20	16	A5C7951597609E2982344D66939AFF	Lina and 6 other went on a tour.
8/1/2017 6:20	8/1/2017 2:20	3	1B7EA1BAFAF72B4EA2AE1ACD8D775F	Lina and 2 other went on a tour.
8/1/2017 6:20	8/1/2017 2:20	4	DDBA7653545B1BB68658838A22BAA5	Lina and 3 other went on a tour.
8/1/2017 6:20	8/1/2017 2:20	5	DF6A386FE701217C2A12292DB8D142	Lina and 4 other went on a tour.
8/1/2017 6:20	8/1/2017 2:20	4	C69E0349776A41A35F9929950CB2FD	Lina and 3 other went on a tour.
8/1/2017 6:20	8/1/2017 2:20	2	349F293BA3435B7BC6AD35BF8396C6	Lina and 1 other went on a tour.
8/1/2017 6:57	8/1/2017 2:57	4	F6DC3043113C54DF6798EBA80C75B8	Mona and 3 other went on a tour.
8/1/2017 6:26	8/1/2017 2:26	2	4E6296831B1C923435A3B21FDAC668	Mona and 1 other went on a tour.
8/1/2017 6:20	8/1/2017 2:20	3	08CD54CB5C552A6A4CBFFDD5D6B326	Lina and 2 other went on a tour.
8/1/2017 6:20	8/1/2017 2:20	2	520EA69D3794B463EC553B79FDF850	Lina and 1 other went on a tour.
8/1/2017 6:20	8/1/2017 2:20	3	9AB43D430EF8C4443FB8698EFD5092	Lina and 1 other went on a tour.
8/1/2017 6:20	8/1/2017 2:20	4	C5F6178AF37CD86DF3B9D8CD811BAB	Lina and 3 other went on a tour.

B- The Unit Test

I created a unit test to check if the results mentioned above (the push notification csv file) are meeting the algorithm expectations or not.
You can find the unit test at (/tester/test.py).

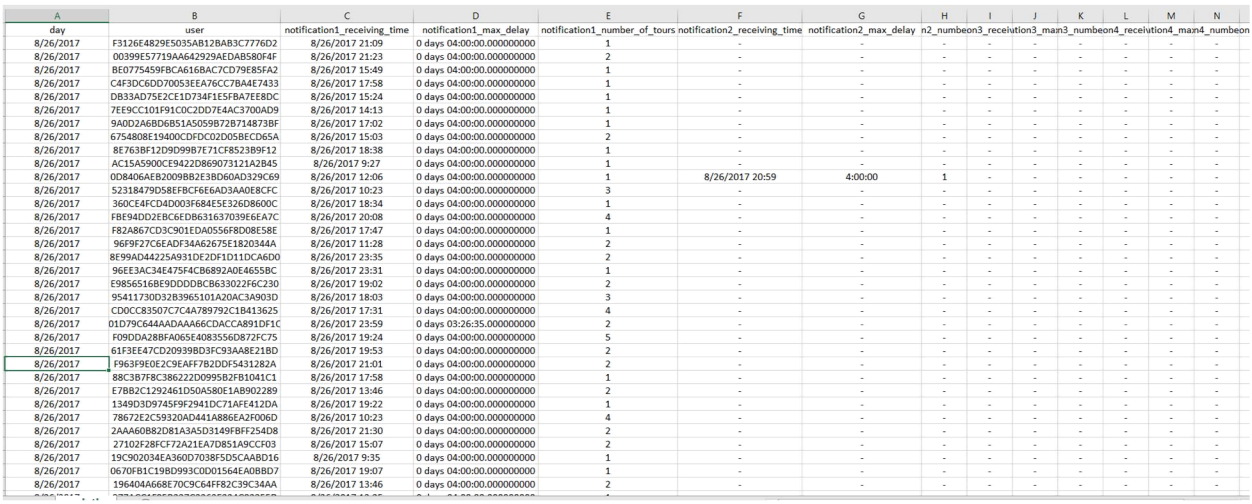
How it works:

- It will ask you to select a push notification csv file like the above mentioned one, in the same format.
- It makes some analysis on this csv file and saves the results to another csv file (/tester/test_results/analytics.csv).

The testing flow:

- It goes through all the push notifications.
- Make data transformation to another data structure (dictionary of date => dictionary of users => tuple of analytics), the analytics are, the first notification time, the maximum delay that this user experienced in this day for each notification during the whole day and the number of tours sent per notification.
- We walk through this analytics dictionary of dictionary of analytics, and save it to csv file.
- In the csv analytics file, you can see these columns:
  - day: the day of receiving the notification.
  - User: the id of the user (it's unique per day).
  - notification1_receiving_time: the time in which this user received a push notification in this day.
  - notification1_max_delay: the maximum delay happened before receiving this push notification (the delay is the time elapsed between action happens and getting it as push notification).
  - notification1_number_of_tours: number of tours included in the above mentioned push notification.

  and the same with all the notifications from 2 to 5 (we agreed that the user gets five notifications maximum per day).



A	B	C	D	E	F	G	H	I	J	K	L	M	N
day	user	notification1_receiving_time	notification1_max_delay	notification1_number_of_tours	notification2_receiving_time	notification2_max_delay	n2_numbeon	3_receiution	3_main	3_numbeon	4_receiution	4_main	4_numbeon
8/26/2017	F3126E4829E5035AB12BAB3C7776D2	8/26/2017 21:09	0 days 04:00:00.000000000	1	-	-	-	-	-	-	-	-	-
8/26/2017	00399E57719AA642929AEDAB580F4F	8/26/2017 21:23	0 days 04:00:00.000000000	2	-	-	-	-	-	-	-	-	-
8/26/2017	BE0775459FBCA616BAC7CD79E85FA2	8/26/2017 15:49	0 days 04:00:00.000000000	1	-	-	-	-	-	-	-	-	-
8/26/2017	C4F3DC6DD70053EEA76CC7BA4E7433	8/26/2017 17:58	0 days 04:00:00.000000000	1	-	-	-	-	-	-	-	-	-
8/26/2017	DB33AD75E2CE1D734F1E5FBA7EE8DC	8/26/2017 15:24	0 days 04:00:00.000000000	1	-	-	-	-	-	-	-	-	-
8/26/2017	7EE9CC101F91C0C2DD7E4AC3700AD9	8/26/2017 14:13	0 days 04:00:00.000000000	1	-	-	-	-	-	-	-	-	-
8/26/2017	9A0D2A6BD6B51A5059B72B714873BF	8/26/2017 17:02	0 days 04:00:00.000000000	1	-	-	-	-	-	-	-	-	-
8/26/2017	6754808E19400CDFDC02D05BECD65A	8/26/2017 15:03	0 days 04:00:00.000000000	2	-	-	-	-	-	-	-	-	-
8/26/2017	8E763BF12D9D99B7E71CF8523B9F12	8/26/2017 18:38	0 days 04:00:00.000000000	1	-	-	-	-	-	-	-	-	-
8/26/2017	AC15A5900CE9422DB69073121A2B45	8/26/2017 9:27	0 days 04:00:00.000000000	1	-	-	-	-	-	-	-	-	-
8/26/2017	0D8406AEB2009BB2E3BD60AD329C69	8/26/2017 12:06	0 days 04:00:00.000000000	1	8/26/2017 20:59	4:00:00	1	-	-	-	-	-	-
8/26/2017	52318479D58EFBCF6E6AD3AA0E8CFC	8/26/2017 10:23	0 days 04:00:00.000000000	3	-	-	-	-	-	-	-	-	-
8/26/2017	360CE4FCD4D003F684E5E326D8600C	8/26/2017 18:34	0 days 04:00:00.000000000	1	-	-	-	-	-	-	-	-	-
8/26/2017	FBE94DD2EBC6EDB631637039E6EA7C	8/26/2017 20:08	0 days 04:00:00.000000000	4	-	-	-	-	-	-	-	-	-
8/26/2017	F82A867CD3C901EDA0556F8D08E58E	8/26/2017 17:47	0 days 04:00:00.000000000	1	-	-	-	-	-	-	-	-	-
8/26/2017	96F9F27C6EADF34A62675E1820344A	8/26/2017 11:28	0 days 04:00:00.000000000	2	-	-	-	-	-	-	-	-	-
8/26/2017	8E99AD44225A931DE2DF1D11DCA6D0	8/26/2017 23:35	0 days 04:00:00.000000000	2	-	-	-	-	-	-	-	-	-
8/26/2017	96EE3AC34E475F4CB6892A0E4655BC	8/26/2017 23:31	0 days 04:00:00.000000000	1	-	-	-	-	-	-	-	-	-
8/26/2017	E9856516BE9DDDDBCB633022F6C230	8/26/2017 19:02	0 days 04:00:00.000000000	2	-	-	-	-	-	-	-	-	-
8/26/2017	95411730D32B3965101A20AC3A903D	8/26/2017 18:03	0 days 04:00:00.000000000	3	-	-	-	-	-	-	-	-	-
8/26/2017	CD0CC83507C7C4A789792C1B413625	8/26/2017 17:31	0 days 04:00:00.000000000	4	-	-	-	-	-	-	-	-	-
8/26/2017	01D79C644AADAAA66CDACCA891DF1C	8/26/2017 23:59	0 days 03:26:35.000000000	2	-	-	-	-	-	-	-	-	-
8/26/2017	F09DDA28BFA065E4083556D872FC75	8/26/2017 19:24	0 days 04:00:00.000000000	5	-	-	-	-	-	-	-	-	-
8/26/2017	61F3EE47CD20939BD3FC93AA8E21BD	8/26/2017 19:53	0 days 04:00:00.000000000	2	-	-	-	-	-	-	-	-	-
8/26/2017	F963F9E0E2C9EAFF7B2DDF5431282A	8/26/2017 21:01	0 days 04:00:00.000000000	2	-	-	-	-	-	-	-	-	-
8/26/2017	88C3B7F8C386222D0995B2FB1041C1	8/26/2017 17:58	0 days 04:00:00.000000000	1	-	-	-	-	-	-	-	-	-
8/26/2017	E7BB2C1292461D50A580E1AB902289	8/26/2017 13:46	0 days 04:00:00.000000000	2	-	-	-	-	-	-	-	-	-
8/26/2017	1349D3D9745F9F2941DC71AFE412DA	8/26/2017 19:22	0 days 04:00:00.000000000	1	-	-	-	-	-	-	-	-	-
8/26/2017	78672E2C59320AD441A886EA2F006D	8/26/2017 10:23	0 days 04:00:00.000000000	4	-	-	-	-	-	-	-	-	-
8/26/2017	2AAA60B82D81A3A5D3149FBFF254D8	8/26/2017 21:30	0 days 04:00:00.000000000	2	-	-	-	-	-	-	-	-	-
8/26/2017	27102F28FCF72A21EA7D851A9CCF03	8/26/2017 15:07	0 days 04:00:00.000000000	2	-	-	-	-	-	-	-	-	-
8/26/2017	19C902034EA360D7038F5D5CAABD16	8/26/2017 9:35	0 days 04:00:00.000000000	1	-	-	-	-	-	-	-	-	-
8/26/2017	0670FB1C19BD993C0D01564EA0BBD7	8/26/2017 19:07	0 days 04:00:00.000000000	1	-	-	-	-	-	-	-	-	-
8/26/2017	196404A668E70C9C64FF82C39C34AA	8/26/2017 13:46	0 days 04:00:00.000000000	2	-	-	-	-	-	-	-	-	-

How the system is expected to be used as a product:

In real life, you would have clock class, which is a simulator for the time now, and you would need to add a flag (notification late fire date/time) which is an indicator to the maximum latency by which you have to fire the push notification, if you have some flag like this, you would be able to say if you have to fire this notification stack for this user at this moment or not.

Maybe we can parallize it too.

5. Results

1. Unit test

Based on the unit test results (I upladed it **here** and gave you edit permission so you can make your own analytics/evaluations online easily), we can see that:

- Average number of notifications per day for each user is 2.5
- Average delay in notifications is 3 hours.

Future work and how can we enhance the algorithm:

Enhancing the algorithm can be done by adapting the ranges more to be user specified, I mean let's assume that the user X used to use the app in the morning, and he/she usually doesn't use it in the evening, so it will be more logical to move the push notifications range to evening, and don't send in the morning at all based on the fact that the user use the app a lot in the morning.

My experience with komoot challenge:

- I like this kind of problems, because they are real life and so interesting, challenging and useful.
- I feel happy to develop such algorithm, this idea jumped to my mind suddenly when I read the challenge document, and I was exited towards it.

- Although the idea is not complex, implementing it and testing it took me time, specially testing it and creating the unit test, because I discovered a lot of bugs and fixed them.
- Overall, the challenge is not hard, it's cool and exciting.
- It took me less than half a day to document my work, and around a day to implement the idea and maybe half a day or less to plan the solution architecture and the flow.
- I learnt that planning is so important, and testing is incredibally important, I solved the challenge around 3 days ago, and didn't create automated testing script, but yesterday I decieded to extract some analytics to see if this solution is really good or not, and found that the results are wrong, because of some logical bugs.
- It's always hard to simulat something, specially the clock, so I would like to know from you how do you usually test such features, all the apps that I worked on so far include direct request/response, but this is the first time-dependent task I work on it, and it's really interesting.