Overview

A publish-subscribe system typically has a set of users, U . Each user has two roles: (a) as a publisher the user is allowed to publish media and as a (b) reader the user is allowed to view media posted by certain other users. For simplicity, only a short text of 30 characters is allowed to be posted.

To implement this project you will have to assign each user a unique user id. You will also have to assign each short text a unique text id. Each short text will also contain a field which will have a flag indicating whether it is a new text, a republished (reposted) text or a reply to an earlier text.

Now let us see the details of the working of the system.

The publishing process: Each publisher (user) is allowed to publish only one short text of up to 30 characters at every time step. This is done through a PUBLISH action. Publication takes place in one of three ways:

1. The user writes a new short text. In this case the text enters the data bases as a tuple: [[time; publishers id; NEW; Text string within quotes]],

e.g., [[86; u27; NEW; “Good morning!”]] is a new short text posted by user u27 at time 86. Once this is entered into the system (posted) it has to be assigned a text id. Let’s say the id assigned to this text is “t1385”.

2. The user reposts an existing text: Format: [[time; publishers id; RE-POST(original text id)]]. Note that this is a text in itself and must be assigned its own unique text id.

Suppose user u14 reposts the text from the example above at time 122, it enters the text database as [[122; u14; RE-POST(t1385)]]. Let’s say this repost gets text id t2277.

3. The user replies to an earlier text: Format: [[time; publishers id; RE-PLY(original text id); Text string within quotes]]. This must also be assigned a unique text id.

Suppose user u31 replies to u14’s report of u27’s tweet at time 136, it is entered as [[136; u31; REPLY(t2277); “What’s so good about the morning?”]].

The reading process Each user subscribes to a set of publishers (who are also users). At any time ‘t’ the user may perform a SUBSCRIBE action to subscribe to a publisher or an UNSUBSCRIBE action to unsubscribe.

To read text posted on the platform the user may perform a READ action.

Let us say the current time at which the user, v, performs a READ action is ‘t0’ and the last time the user performed a read was at time ‘t −1’ < t 0 then a list of texts are returned. A text x is included in the list

1. If the user who posted x is u x and v subscribed to u x at time t s < t 0 and did not unsubscribe from u x between the t s and t 0 and
2. if the timestamp t x of x is such that max{t −1 , t s } ≤ t x < t 0 .
3. Or if t −1 ≤ t x < t 0 and x is a reply to some text posted by v.

In other words the text should either be a reply to one of v’s text from any user or it should be an unseen text from a publisher that v is currently subscribed to.

The texts should be output sorted by time. Before a user has performed any READ we can assume that t −1 has some default value like -1 which is
smaller than any allowed timestamp.

Actions:

• PUBLISH,t,uid,NEW,text,tid: User uid publish new text at time t having text id tid.

• PUBLISH,t,uid,REPOST(ptid),tid: User uid repost existing text represented by text id ptid at time t. The repost text is assigned a new text id tid.

• PUBLISH,t,uid,REPLY(ptid),text,tid: User uid reply to an existing text represented by text id ptid at time t. The reply text is assigned a new text id tid.

• SUBSCRIBE,t,uid,pid: User uid subscribe to a publisher pid at time t.

• UNSUBSCRIBE,t,uid,pid: User uid unsubscribe to a publisher pid at time t.

• READ,t,uid: User uid reads text at time t.

Points to note:

- In the input files, every line will be a new ‘action’
- The action that comes later will be happening later in time