Assumptions:
- Each PROCESS works as Reporter.
- The process with rank 0 is the Editor.
- Reporters communicate with each other when they are working on the same topic.

1.Standard(BNPS)
Model:
1.Same files having news containing topics and their updates are provided to the reporters.
FILE STRUCTURE:-

```
  Topic            //STRING
  Valid/Invalid     //INT: this bit would be checked by editor at the end to ensure that this topic
is not repeated twice.
  U1               //STRING: Updates are arranged in increasing order of their timestamp, hence
                             no explicit timestamp is provided.
  u2
  .
  .
  .
  Un
```

2.Reporters/processes, will pick topics from the file at random from the files, and updates in random but increasing order of their timestamp from the file.

3.Each reporter has a data structure storing the information about who is the leader of the every topic, which is initialized to -1 initially.

4.The reporter who is the first to hit the topic is assigned the leader of that topic and it forms the point of communication between reporters working on the same news topic.

5.Whenever a reporter comes across a news topic and its update it will check its data structure if there is a leader for that topic. If this value is found to be -1, then it will enquire about the same from the editor.

6.Editor assigns the first person to open a particular topic, as its leader, and any further enquiries about the topic are redirected to its leader.
7.Suppose now that the reporter knows about the ID of the leader for a particular TOPIC, it will communicate with that process to check if any other reporter is working on the same topic as his, if yes it will skip the update and go for next report.

ABOUT THE LEADER OF A TOPIC:

8.As soon as a reporter becomes the leader of a TOPIC, it will spawn a new thread to serve as the communication medium between reporter, whereas the original thread continues the work of a reporter. One process can be the leader for multiple topics.

9.The reporters will send the updates to these leaders and leaders will store only the latest update.

10.Each reporter after finishing its task will, send a termination message to the editor, hence at the end of the day, editor communicates with the leader of each topic to fetch the latest update available with them.

THIS IS JUST TO SIMULATE THE REAL LIFE PROBLEM
11.It then runs a validity test, to check if the same news has already been telecasted, before finally passing it for LIVE TELECAST.

2.Multiplex(BNPM)
MODEL:

1. BNPM problem is solved for 2 cases: fixed reporters for each editor and varying reporters for each editor.
2. Any reporter process communicates with its editor first to get the leader of the topic on which it wants to work.
3. Editor gives the leader of the topic and now the reporter communicates with the leader of the topic to ensure that only one reporter works on a particular update.
4. Now all the editors communicate with the leaders of each topic to get the latest update of each topic.
5. All the editors communicate with editor of rank = 0 process. Let's call it Editor0.
6. Each editor give the topics and latest updates to editor 0. Editor 0 will telecast the topics with latest updates as the breaking news.

Q2. (i) Derive an expression for the number of MPI calls for a given number of reporters and a given partition of them.

Let editor = e, reporters_per_editor = r, topics = t, no. of updates for each topic = u.
For BNPS problem e = 1, for BNPM problem e > 1.

In the worst case, Number of MPI calls is given by the following formula, for a particular telecast:

$2 * ((t * r) + (t * u * r) + (t * 1) + ((e - 1) * t)$

Each term is described below:

t * r : Each reporter calls the editor to work on each topic (number of topics = t).

t * u * r : now all the reporters(=r) will ask about all the updates (=u) to their group leader (= t) .

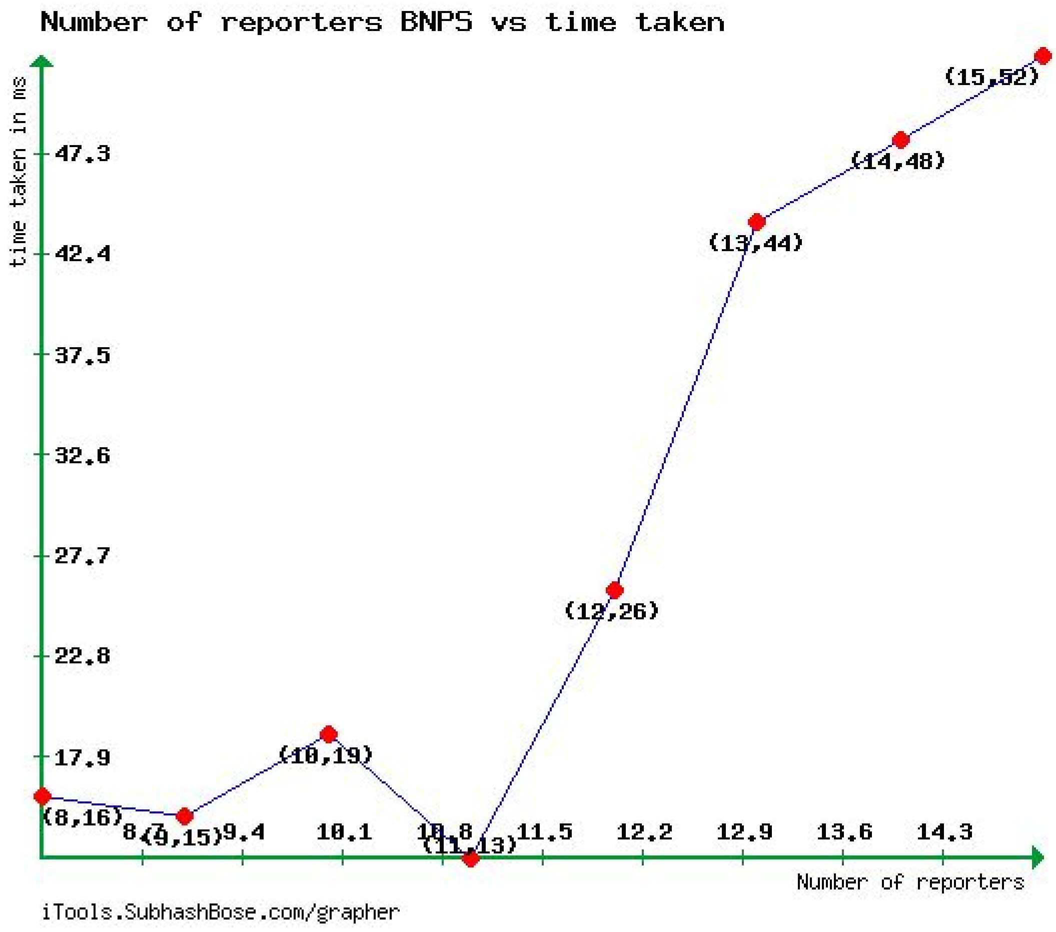
t * 1 : the leaders of respective topics(=t) give the latest update of each topic to editor.

(e - 1) * t : this term is 0 for the BNPS problem, since e = 1 for that. In BNPM, all the editors except editor 0 (= e-1) will send the latest updates for all topics (=t) to editor0, and editor0 telecasts the latest update of them all as the breaking news.

Finally multiplication by 2 is because for send call, there is also a recieve call.

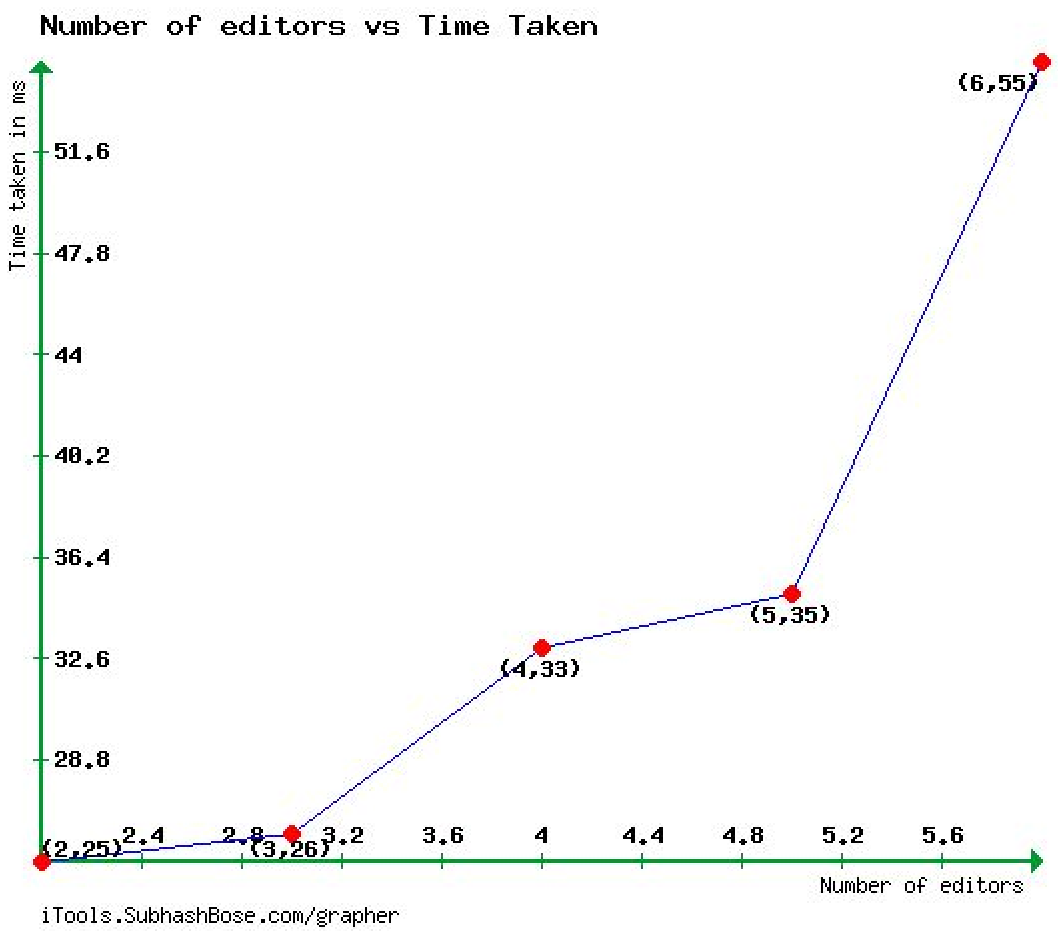
Q2 (ii) Please see the attached TimeAnalysis file.

Q2(iii) (a)

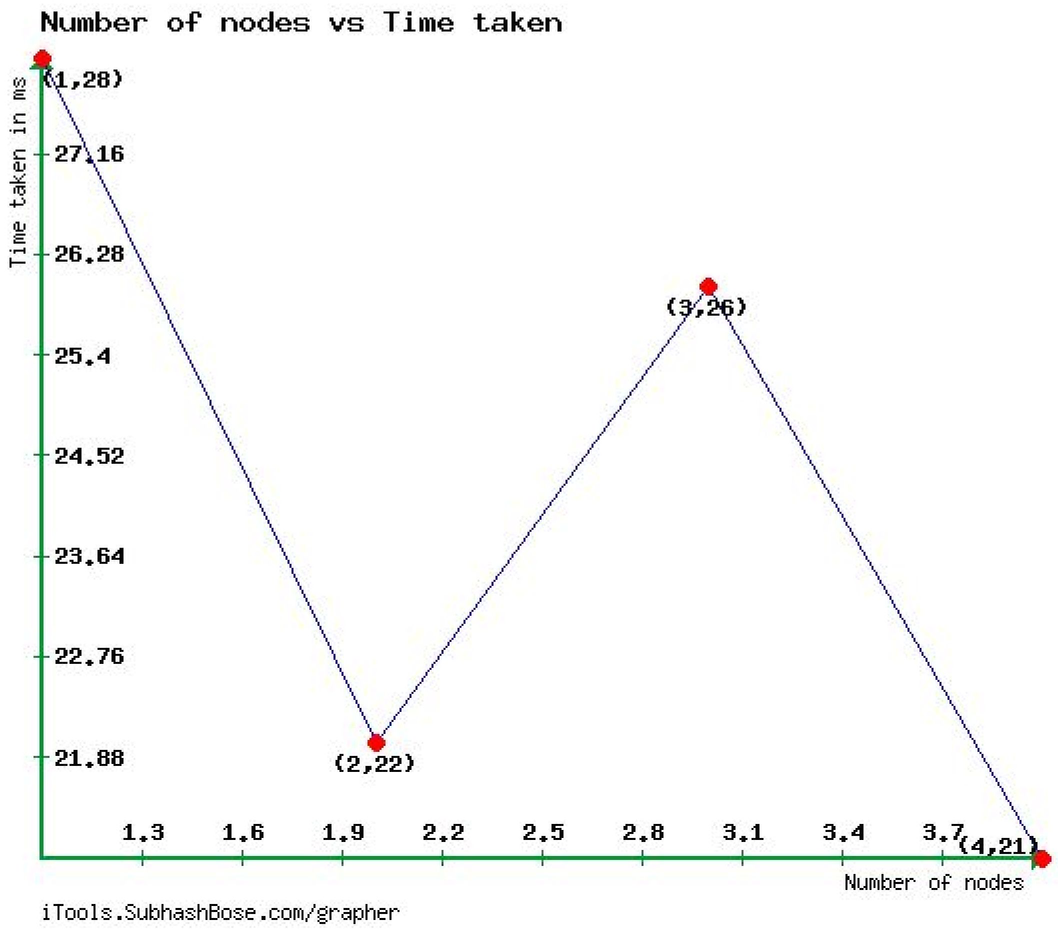


(b)

For number of reporters per editor = 2, following is the graph obtained:

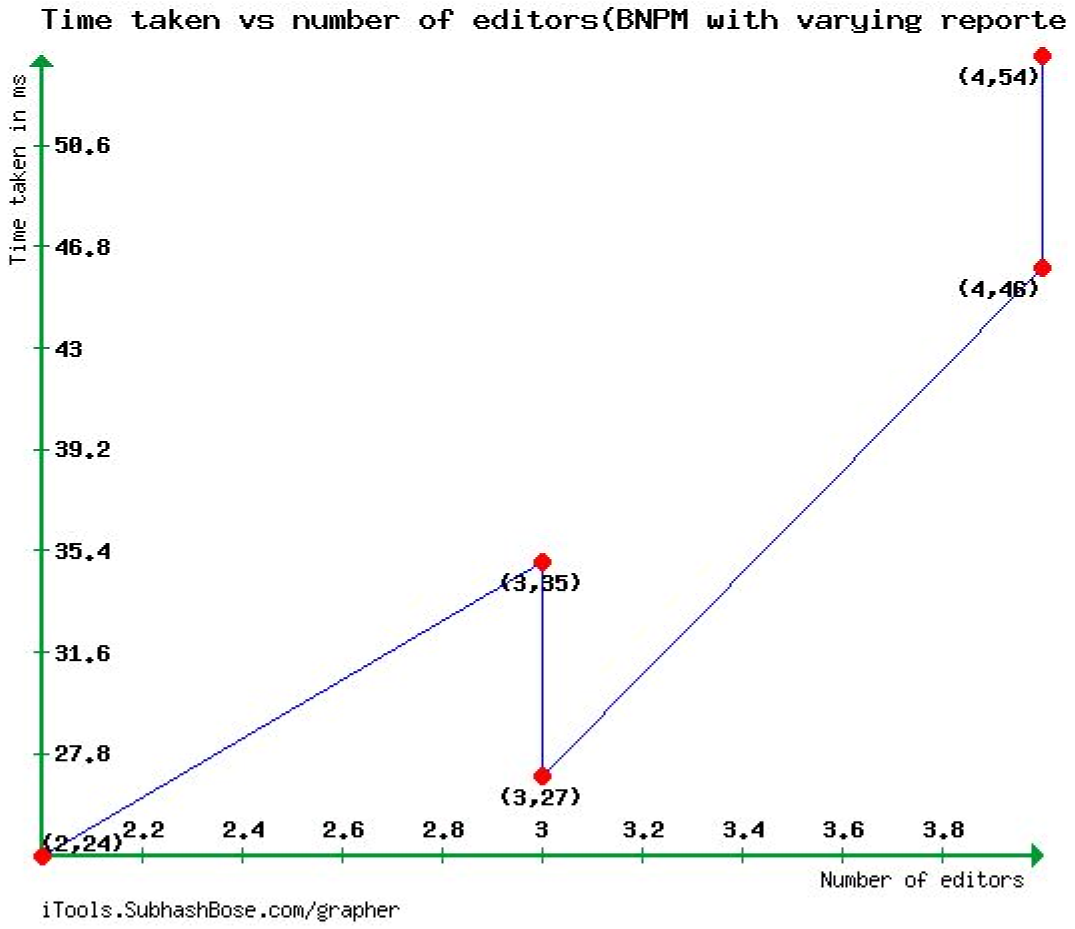


(c) BNPS. Number of reporters = 10, Editor = 1. Following is the graph obtained:



(d)
Time taken vs number of editors(BNPM with varying reporters)
Number of cluster nodes used = 4

Number of editors	Number of reporters	Time taken(seconds)
2	1 3	0.024
3	3 2 1	0.035
3	4 1 1	0.027
4	1 1 2 4	0.046
4	3 3 1 1	0.054



Q2 (iv) Our solution is very scalable as no node is getting overburdened with the amount of reporters it needs to process. Our idea that the communicate with the reporter_leader of the particular topic ensures that we only take n mpi calls for communicating in a group of n reporters. Also since there can be different leaders for different topics, so we ensure that a particular reporter leader manages only a small set of reporters , as will occur in practical real life situations. So, even if we increase the number of reporters to an arbitrarily large size, the load will get distributed among various reporter_leaders and there will not be a single node which will become a bottleneck. In the BNPM problem also same strategy is adopted. So as can be seen from the graphs in part (iii) above, the time taken does not increase to arbitrarily large even if we increase the number of reporters or number of editors.