Project report
Reliable news over a gradient topology

Gautier Berthou

Michael Eusebe

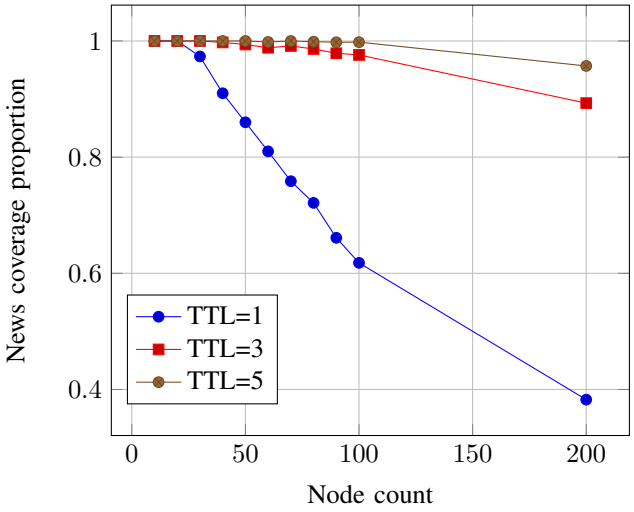

Fig. 1. Influence of network size and TTL on coverage

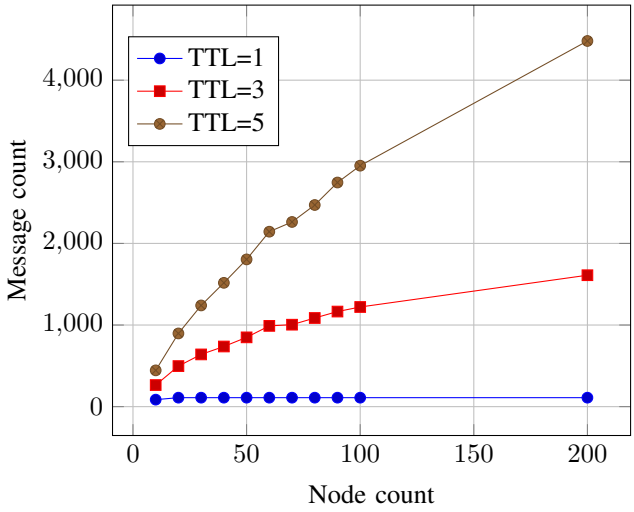

Fig. 2. Influence of network size and TTL on traffic

I. Task1 - News Flood

In our code, the number of nodes can be controlled by modifying the constant named `ScenarioGen.NETWORK_SIZE`, the number of news to be sent during the simulation is set by `ScenarioGen.NEWS_MAXCOUNT` and the initial TTL of all peer-sampling messages is set by `NewsFlood.INITIAL_TTL`.

Throughout the whole experiment, the number of news was set to 10 in order to compute our data based on the average results over 10 news. We also ensured that simulation time was long enough to let all peer-sampling messages reach TTL=0, which means that at the end of the simulation all the news reached their maximal propagation over the network.

Figure 1 shows how much impact have network size and TTL value on news coverage. Scale is from 0 (no node knows the news) to 1 (all the nodes know it). It can be seen that as we expected, it gets harder for a news to propagate onto the network when it gets more nodes. In addition, it appears that a TTL value of 1 is not enough to propagate efficiently a news. For example, in a 100 nodes network, only 61.8% of the whole network knows the news in average. When TTL value is increased, higher news coverage can be achieved. Average node knowledge follows exactly the same distribution as the average news coverage.

Figure 2 shows the influence of network size and TTL value on traffic. As in our simulation all messages have the same size, only the number of messages transferred between nodes becomes relevant to study. As we expected, the number of exchanged messages increases when TTL increases or when the network grows. Given a static network size, the number of messages increases quadratically as a function of TTL value, which implies that high TTL values will quickly generate numerous messages. As the number of exchanged messages increases, news coverage also increases (*cf.* figure 2), but there are also many messages which will not lead to any improvement. Indeed, many of them will be sent to nodes that already know the news so there will be a significant waste over the network, depending on the TTL value and network size.

In conclusion, when the target network is expected to be always rather small, small values of TTL (3 seems to be a good compromise between traffic generation and coverage) are efficient enough. However, when trying to scale up the network, bigger TTL values must be used. Alternatively the peer-sampling protocol might be modified to optimize coverage without increasing the TTL value.

II. Task2 - Leader Selection

In figure 3 we can see the linearity of the number of rounds that are necessary to stabilize the network. Stability is reached when the gradient component provides the same neighbors a certain amount of rounds a row.

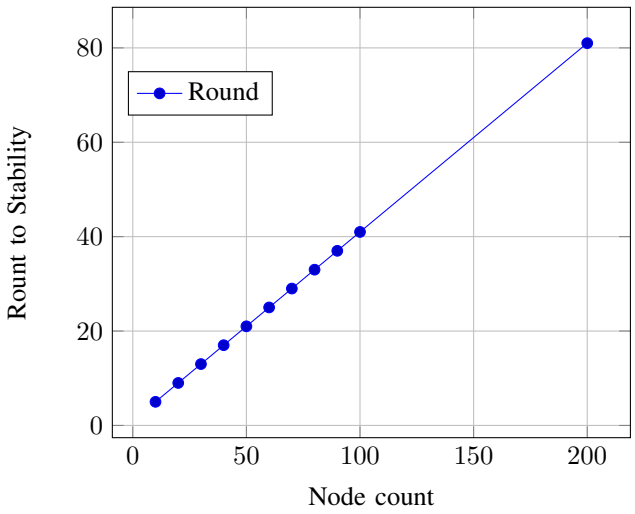

Fig. 3. Influence of network size on the amount of necessary rounds to stabilize the network

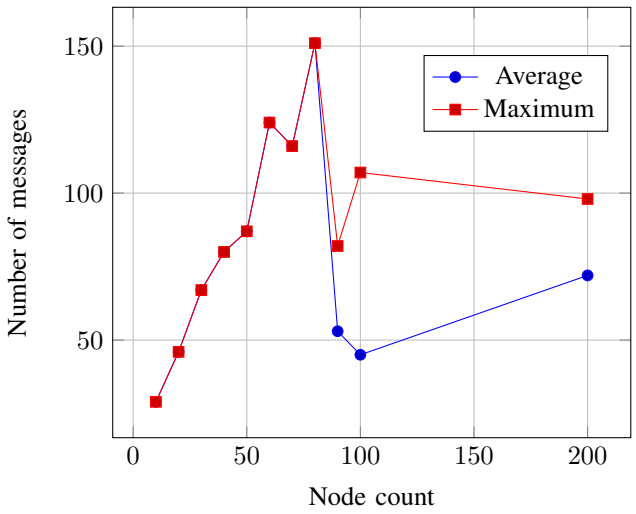

Fig. 4. Influence of network size on the number of necessary messages to select a leader

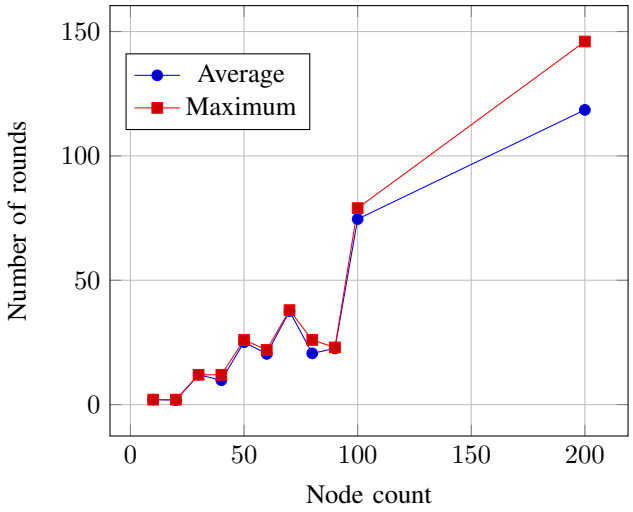

Fig. 5. Influence of network size on the number of rounds to propagate a news summary

After having stabilized the network a leader has to be selected. To begin selection every stabilized node compares itself with its gradient neighbors. If one of them finds out that it is superior to all its neighbors, it might be leader and verification begins.

To verify if a node is the leader, it will ask its neighbors to make the same comparison with their own data. And to be completely certain of it, all nodes will save the address of the leader candidate and send their own neighbors to it. The actual leader candidate will compare the nodes it has already checked with the ones it received and will do the same as with its neighbors with the nodes that are not yet verified. This process will continue until all center nodes have been verified. Because neighbors are the closest nodes to a given node with a preference for higher nodes, message exchanges will only happen in the center of the network. If there is a conflict, that is to say a leader candidate encounters at least one superior node, it will simply send to the new leader candidate all nodes it has already checked and the algorithm will continue. When the last node to be checked sends a list with only formerly checked nodes, the leader is elected: the leader candidate becomes the actual leader.

Figure 4 shows the evolution of the amount of necessary messages to select the leader according to the number of nodes in the network. Message count increases until network reaches 90 nodes. Then it decreases and finally stabilizes to a value between 50 messages and 80 messages. The first part of the curve increases because there are more nodes to be verified to ensure that the leader candidate is the right node. At 90 nodes, the network gets big enough to develop a center, that is to say a group of nodes not directly connected to the edge of the network, thus the number of nodes to reach decreases. At 90 nodes in the network we measured 1700 messages. The corresponding point is not shown on the graph to keep a clean scale and because it looks like an outlier. After having reached the point where a center can be formed, the number of necessary messages is quite constant because the gradient topology keeps approximately the same amount of nodes at the center.

III. Task3.1 - Leader Dissemination

It can be seen from figure 5 that the number of message rounds actually involved in the news summary dissemination from the new leader is rather high compared to what we would have expected from taking advantage of the gradient structure of the network. However, these results can be explained by the fact that the messages do not keep any history of the nodes the message has already been sent to. As a result, a significant amount of messages are transmitted to nodes that received the news summary earlier. Even if such nodes will ignore the message, it is still accounted as a message round. On another

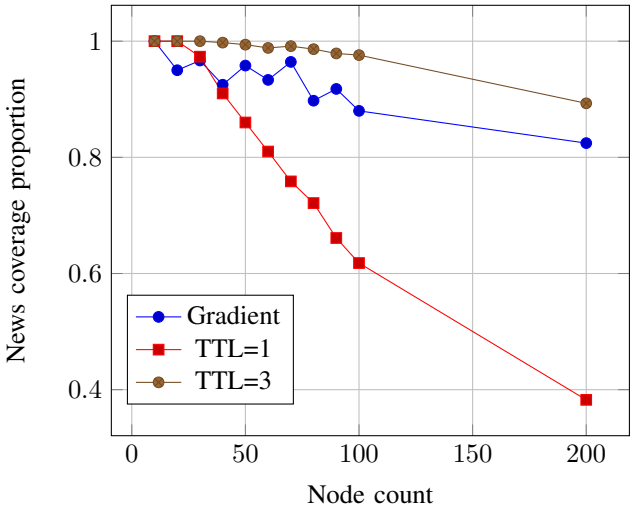

Fig. 6. Comparison of news coverage

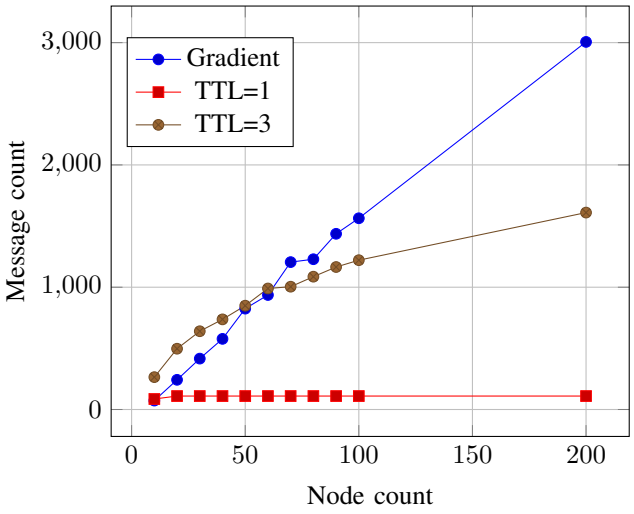

Fig. 7. Comparison of traffic

hand, average and maximum values are close when network contains less than 100 nodes.

IV. Task3.2 - News Dissemination

Figure 6 shows a comparison of news coverage rates when using a simple TTL-based peer-sampling newsflood algorithm or when using a gradient-based algorithm. The latter gives better results than a TTL-based algorithm with its parameter set to 1 for any larger network than 40 nodes. However, a TTL-based algorithm with its parameter set to 3 is better than the gradient-based one. The way the gradient component computes the fingers and the neighbors of each of the nodes may be a cause to the observed results because it is the only information they rely on when propagating a news summary. The dissemination algorithm itself can also explain our results.

Figure 7 shows a comparison of the amount of messages that were generated during the news dissemination process between a simple TTL-based peer-sampling newsflood algorithm and a gradient-based algorithm. A TTL-based algorithm which parameter was set to 1 gives the least amount of traffic, but its news coverage was the worst in most cases so it is not worth using it when the network contains more than 40 nodes. The gradient-based algorithm generates less traffic than the TTL-based one with a TTL value of 3 when network size is less than 50. For networks that contain more than 50 nodes, the gradient-based algorithm has the biggest impact in terms of traffic. This can be explained by the fact that any TTL message will eventually die. So, when the network is small enough (here, less than 50 nodes when TTL=3), there is an abundant message redundancy when using TTL. When the network is big enough (more than 50 nodes when TTL=3), the fact that the messages actually die reduces redundancy. On the contrary, the messages sent by the gradient-based algorithm do not contain such a hop counter and they are not attached any node history (like news summary dissemination) so the algorithm hardly knows when to stop sending messages.

Our algorithm relies on a push protocol. As soon as the leader has a new news, it pushes it to its neighbors within the gradient topology. When a node receives such a news, it first checks whether or not it already knows it. If it already knows it, the message is ignored. Otherwise it is forwarded to its neighbors. Although our algorithm relies on data that was provided by the gradient component, its properties are not very good when trying to keep the protocol as simple as possible (no nodes history). A pull algorithm might have generated less traffic because the nodes might pull several news at the same time. In addition they only need to send their pull request to fewer nodes (ideally only one higher rank node) so it would avoid much redundancy. However a news would take a bigger time to be known by all the nodes because of the periodic pull mechanism.

V. Task4.1 - Leader Dissemination

We were not able to make the node kill feature work as we expected so we did not get any result to base our analysis on.

However the selection protocol has been improved to support dynamic node failure and it performs leader failure detection. To achieve failure detection, the leader, after its election, sends its address to its neighbors. All of them will check its availability every 10 gradient rounds. If the leader did not answer during few rounds, a new leader selection is launched. During the selection every node can fail. If the leader candidate keeps the same list of nodes that need to be verified for a few rounds, it assumes that these nodes have failed so it clears the list to elect the leader.

These enhancements have a very small impact on the gradient stabilization protocol and on the leader selection algorithm.

VI. Task4.2 - Leader Dissemination

We did not manage to realize this task because of the same reasons we did not get reliable data on task 4.1.

However we have thought about how push/pull mechanism would behave in presence of churn. In both cases the presence

of churn makes the protocol much less deterministic. The fact than we cannot predict what will happen is a big issue.

VII. Annex

The results shown in this document were generated throughout the development. As a result, running again the simulations might output different results.