ON THE MODELING OF P2P SYSTEMS AS TEMPORAL NETWORKS: A CASE STUDY WITH DATA STREAMING

Luca Serena

DISI, University of Bologna
Via dell'Università 50 - Cesena, ITALY
luca.serena2@unibo.it

Mirko Zichichi

OEG, Universidad Politécnica de Madrid
ETSIINF, Boadilla del Monte (MD), SPAIN
mirko.zichichi@upm.es

Gabriele D'Angelo

DISI, University of Bologna
Via dell'Università 50 - Cesena, ITALY
g.dangelo@unibo.it

Stefano Ferretti

DiSPeA, University of Urbino "Carlo Bo"
Piazza della Repubblica, 13 - Urbino, ITALY
stefano.ferretti@uniurb.it

Abstract

Temporal networks are a useful tool to model complex systems' dynamics, especially when they are characterized by high dynamicity. While there is strong literature on simulation tools for complex and dynamical systems, there is a lack of viable solutions to model and exploit temporal graphs in simulation. In this work, we present a system devised to simulate complex systems and their evolution by using temporal graphs data structures. As a use case, we focus on data dissemination over peer-to-peer systems characterized by a relevant presence of churns. The simulation of dissemination algorithms on temporal graphs involves evaluating their efficiency in terms of coverage, delay, and number of messages sent. In particular, a reasonable trade-off between the speed of delivery and the generated network traffic must be found. In this work, besides traditional gossip strategies, a more complex scenario is considered, where a second overlay, structured as a tree, is built for more efficient propagation of data from the source to the interested peers only. This scenario is analyzed using multiple simulated strategies. More specifically, we investigate how the simulation methodology can be used for evaluating the efficiency of dissemination protocols in a peer-to-peer environment but with a specific focus on the modelling of churn.

Keywords: simulation, temporal networks, peer-to-peer, dissemination, gossip.

1 INTRODUCTION

Complex systems are quite often modeled and simulated as a set of entities that somehow interact in an environment. Interactions are made possible among those entities that share some specific properties (e.g., they are at a short distance, given whatever definition of distance). The possibility of interaction between two entities is often represented as a link between the two entities, thus creating an interaction network. The logical connections among the entities involved, depending on the nature of the system, might be subject to changes over time. That means either that (i) new entities enter the system at some point while other entities may disappear, or that (ii) the links between entities are not stable and can change over time. Thus, upon the need to model and study the dynamics of such complex systems, modelling and simulation techniques need to consider the temporariness of the links, which can actually be relevant for the outcome of the tests.

Specifically, in our paper, we employ temporal networks in order to simulate a peer-to-peer (P2P) streaming scenario, where multiple nodes (i.e. downloaders) listen to a stream of data emitted from a specific peer (i.e. the streamer node). In a P2P scenario, the nodes form an overlay network (i.e. a logical network built on top of another network, like the Internet), and the structuring of the overlay changes over time, due to new participants joining the system and other nodes leaving the network. The purpose of this paper is to propose a methodology and a simulation tool that enables the analysis and the comparison of different strategies for streaming-like applications. Modelling and simulation techniques permit investigating the impact of certain strategies and parameters on the performance of the application, analyzing also how the dynamic topology of the overlay affects the functioning of the system. For example, the delay experienced for messages delivery and the amount of network traffic are big concerning issues for P2P systems, and it is desirable to find an appropriate trade-off, finding solutions that enable the reduction of the number of messages sent while not increasing too considerably the average delay. Due to the volatility of the participants, we think that temporal graphs are an appropriate data structure for the modelling of this kind of system.
Through the use of LUNES-Temporal, a simulator specifically designed for distributed systems, we were able to model the temporal graphs and the P2P protocols, focusing on the overlay aspects of the system. Specifically, the most relevant aspects considered in the modelling of the system are: (i) the usage of a temporal graph for the adequate representation of churns, (ii) different weights in terms of messages size for a precise estimation of the overall network traffic, (iii) information about nodes obtained only through the exchange of messages, (iv) hops as the measure for expressing the delay, according to a time-stepped simulation approach, where a time-step represents an atomic unit of time (v) possibility of nimbly changing network and protocols parameters in order to evaluate their impact.
In this work, we want to display some techniques for modelling dynamics systems through the use of temporal networks. We show that it is possible to create efficient and scalable simulators, which allow the users to carry out different experiments by easily customizing the behaviour of the agents and the network parameters. The remainder of this paper is structured as follows. Section 2 provides the background and a discussion of the related work. In Section 3 the tool used to model the overlay network and for the conduction of the experiments is introduced. In Section 4, the collected results are discussed. In Section 5, the performances of the simulator under different conditions are reported. Finally, in Section 6 the concluding remarks are provided.

2 BACKGROUND AND RELATED WORK

In this section, we introduce the background that is necessary for the rest of the paper. In particular, we add some details about the modelling and simulation of temporal graphs and we briefly describe P2P systems and some of the possible strategies for spreading data among the participants. Moreover, we discuss the most relevant related works.

2.1 Temporal Networks Modeling and Simulation

Temporal graphs represent the interactions among entities over time and, unlike normal graphs, the set of nodes and edges is subject to variations as time advances, enabling to capture the impact of temporariness. An area where there are plenty of simulation investigations that make use of temporal networks is the study of epidemiological scenarios, where temporal graphs are used to model human interactions (Antulov-Fantulin et al. 2015) (Riad et al. 2019). Also communication systems can be analyzed with the help of temporal networks, for example, in (Hiraoka et al. 2020) a model is proposed to study bursty behavior in the activity of certain nodes. On the other hand, temporal graphs can also be employed to analyze distributed systems like P2P applications, where participants and their connections continuously change over time. The temporariness of a graph can be a more or less influential factor depending on the simulated architecture. For example, in a tree-structured dissemination scenario churns (i.e. arrivals and departures of peers in the

system) play a big role in the performance, unlike when dissemination is only based on a P2P overlay, as shown in (Serena et al. 2021). A possible strategy for the modelling of a temporal network, since the number of nodes in the graph is not fixed, but continuously subject to variation over time, is to (i) set the total number of simulated entities, (ii) set a percentage of nodes that are active, and (iii) set the percentages for the activation and for the deactivation of the nodes. LUNES-Temporal has been built according to this approach.

2.2 P2P Systems as Temporal Networks

Peer-to-peer systems are usually built on top of an already existing physical network (Malatras 2015), like the Internet, and the underlying overlay network can be represented as a graph (directed or undirected depending on the specific implementation), where the nodes are the peers of the system and the edges connect the various neighbors (i.e. the nodes that are directly in touch). In particular, to model distributed environments characterized by the presence of churns, we can make use of temporal graphs. In typical P2P systems, the participants rarely remain connected for more than a few hours, and they frequently exit the system after only a few minutes (Cohen 2003). It is evident that churn can lead to a more complex overlay management with respect to static networks. In fact, certain policies must be adopted to avoid issues such as nodes isolation, which occurs when all the neighbors of a node n fail before n is able to detect their departure and to replace them with other alive peers (Yao 2009). Usually, the overlay networks are classified according to two categories: tree-based and mesh-based networks (Neysiani et al. 2012). In tree-based P2P networks, the source node (that is the node that is initiating the dissemination) is placed as the root of the tree and its neighbors are assumed to be child nodes. Each child node is the root of a sub tree and this is defined recursively. From the source node, the messages are propagated following a top-down scheme, eventually reaching the leaves. In certain circumstances, this approach can be very efficient for P2P dissemination. In fact, it is possible to push the streaming content only once to the interested nodes, thus optimizing the number of forwards. However, the construction and the management of the tree might be a costly operation due to the presence of churn. Every time that a (non-leaf) peer exits the system the tree breaks, and besides, a logical tree is necessary for every source of data.
On the other hand, with a mesh-based approach, the participants form a randomly connected overlay, trying to maintain a certain number of both incoming and outgoing connections (Magharei et al. 2007). Upon entering the system, each peer sends a join message to a tracker (or a bootstrap node), which provides a list of available peers with whom it can set up a connection. These connections will be used for pushing content and for pulling requests. A relevant advantage of this approach is that messages can travel from source to destination through multiple paths, with more tolerance to peers' failures. Furthermore, no structural constraints prevent nodes from contributing to the spread of data, such as leaves in tree-based overlays (Pianese et al. 2007).

2.3 Dissemination Protocols

In a P2P environment, several strategies can be adopted to diffuse content among the participants. Their feasibility and efficiency specifically depends on the architecture and on the features of the system. Often, a node that is sending a message does not know the address of the recipient, thus a gossip protocol is employed to dictate the policy for data dissemination. There exist three main approaches for the nodes to set up dissemination in a P2P content sharing system: (i) *push* algorithms where data are spread into the network as soon as they become available, (ii) *pull* algorithms, where the content is disseminated only when it is explicitly required and finally (iii) *hybrid* mechanisms where the previous two approaches are combined together. Usually, a hybrid strategy may reveal to be convenient, for example allowing content owners to push the information into the system, with the possibility for the recipients to ask for the missing

information, according to a pull-based approach (Cigno et al. 2008). Independently from the direction of the messages (whether from the source to the downloader node or vice versa), several strategies for disseminating the information exist.
A first idea, that is in theory optimal for minimizing the delay, is to always broadcast the data received for the first time toward all the outgoing connections. This approach (i.e. typically referred to as full broadcast), though, is very costly in terms of network traffic, in particular when the average degree (i.e. number of neighbors of each node) is high. Therefore, it might be desirable to adopt other gossip protocols, which are able to ensure an appropriate trade-off between the number of messages sent and delivery time. Some pull-based algorithms are proposed in (Ouali et al. 2009), considering the maximum upload bandwidth of the neighbors, while other gossip protocols aim at minimizing the network traffic by forwarding the messages only to a subset of neighbors, according to some policy. In particular, the following algorithms have been proposed:

- *Probabilistic Broadcast* (PB). Given a forwarding parameter p, there is probability p that a node forwards the message to all the neighbors except the forwarder and probability $(1-p)$ that it does not send it to any other node.
- *Fixed Probability* (FP). Given a parameter p, a roll of dice for each of the neighbors (except the forwarder) takes place, so each neighbor will have $p\%$ chance of receiving the message.
- *Degree Dependent Function Algorithms* (DDF). The message is sent based on the degree (i.e. number of neighbors of a node) of the potential receiver: the less connected is the neighbor, the more possibilities it has to receive the message (D'Angelo and Ferretti 2017).

Independently from the protocol being used, a cache mechanism and a Time-To-Live (TTL) are employed to further reduce the dissemination overhead and to prevent infinite loops of messages. The strategy is respectively to not forward an already received message and to allow a message to be relayed only for a limited number of hops.
The metrics that are typically used for evaluating the performance of such algorithms are the *chunk reception rate* (i.e. the percentage of chunks that the downloaders are able to receive while they are listening to the stream), the *coverage* (i.e. the percentage of active peers that receive messages during the dissemination), the number of *messages sent* throughout the execution of the protocol and the *delay* (i.e. the average time required to contact the recipient nodes)

3 LUNES-TEMPORAL

LUNES (Large Unstructured Network Simulator) is a time-stepped discrete event simulator designed for the simulation of complex networks. In this paper, we present LUNES-Temporal, a specific version of LUNES that allows for the simulation of temporal networks (lun 2022). The software is built on top of ARTÌS/GAIA (D'Angelo 2017) simulation middleware, which offers the primitives for time management and communication among simulated entities, enabling also for parallel and distributed executions. The tool has a big focus on scalability, allowing for the modelling of systems with a large number of simulated entities. In the simulation model, the agents communicate through the exchange of messages: at each time-step every simulated entity receives messages from other agents (if any) and then performs some actions, potentially sending in turn messages to other simulated entities.
The simulator has been designed to be easily extendable and customizable, in order to implement and test various types of protocols. First of all, it is possible to tailor the network topology, which can be treated like a graph where the nodes are the simulated entities and the edges are links between two nodes. In the case of temporal networks, the graph structuring is very volatile, and some important decision parameters are the average number of connections that a joining node is establishing (in our case a connecting node

chooses a variable number of connections ranging from a lower bound to an upper bound) and the strategy for managing churns (e.g. preventing a node from remaining without active connections).

4 RESULTS AND ANALYSES

The evaluation of a simulator for temporal networks involves two different levels of assessment. First, we need to understand if the simulator allows us to effectively represent and study the use case we adopted. This serves to understand if LUNES-Temporal allows for creating viable and easy-to-use simulation models. To this aim, in this section, we study different metrics related to the dissemination strategies over the temporal graphs modeling the dynamic P2P systems. Second, we want to assess the performance of LUNES-Temporal in terms of execution speed (i.e. time required to complete a simulation run). The results about the simulator performance are discussed in the next section.

4.1 Content Delivery Protocol

Following the idea proposed in (Zhang et al. 2007), we consider a hybrid push-pull protocol, which encompasses both mesh-based and tree-based approaches for data dissemination. The rationale is to have two separate overlay levels for the nodes, the mesh-based for pulling the requests for a stream and the tree-based for pushing chunks. This approach enhances the decentralization of the system by not relying on servers (or special participants) when discovering which peers own certain resources.
The protocol is composed as follows. When node A wants to join a data stream, it disseminates a Request message. The IP address of the streamer is not known, thus a pull-based gossip algorithm is employed to contact the streamer. Once the streamer S receives a request from node A, it adds A to the stream tree. Assuming that every node can relay the data to at most n other peers (i.e. n is the maximum degree for the nodes), if S has fewer than n children, then S adds A as a child. Otherwise, S selects randomly one of its children, which will be in charge of placing A in the streaming tree. The procedure will proceed recursively until a destination for A is found (i.e. A becomes a leaf of the tree). Finally, after waiting for a short amount of time (that is needed for the tree construction), the streamer starts emitting data. Other nodes can dynamically join the streaming tree at any point during the execution. If a non-leaf node belonging to the streaming tree becomes inactive, its children will send again the request message through the overlay once detected the failure, in order to join the streaming tree again.
This strategy aims at minimizing both latency and network traffic, which are two of the most concerning issues in P2P environments. In fact, the only messages that are spread into the network via dissemination are the request messages, which only contain a few data (i.e. only the information about the node itself and about the stream to join). The largest load of data (i.e. the chunks) are forwarded exactly once for each participant of the stream and they employ at maximum h relays to reach a participant node, where h is the height of the unbalanced tree. This means that typically the chunks account for the largest part of the communication overhead in this kind of system.

4.2 Setup and Methodology

In our simulation, the time is divided into epochs, with each epoch being characterized by a single source of data (i.e. before starting an epoch a single active node is elected as streamer). In each epoch, the streamer emits 470 chunks, one at each time-step. In the tested configuration, the system is composed of 10000 simulated entities, among which 80% on average is active, and every active peer has on average 10 connections in the mesh-based overlay. Among the active nodes, 5% of them take the role of the downloader, and in turn, the nodes switching from deactivated to active in the middle of an epoch will have the same probability to also apply for the stream. We also assume that the streamer is waiting for 10 time-steps at the beginning of

an epoch before starting emitting chunks, in order to wait for the downloaders to arrange in the streaming tree. In our model, the simulated entities are either active or deactivated, and there are three types of active nodes:

- **Non-Downloader** nodes, which are part of the overlay network like all the active nodes. They have a variable number of neighbors, contributing to the dissemination of messages.
- **Downloader** nodes, which are a subset of active nodes that are also interested in receiving the data emitted from the streamer. They first send request messages, in order to be added to the steaming tree. Then, once received an item message, they relay it to the children node if they have any.
- **Streamer**, which is the node that generates the stream of data and is also the final recipient of the request messages. It is the root of the streaming tree and it spreads the information to the graph by forwarding the various chunks to its children nodes.

The interactions among the nodes are based on the exchange of messages. More specifically, five different types of messages have been employed:

- **Request** messages, which are sent by the downloaders in order to listen to a stream of data at the beginning of an epoch or when they join the system. These are the only messages spread via dissemination with a certain gossip algorithm.
- **Item** messages, which contain the chunks generated by the streamer and requested by the downloaders. They are sent through the streaming tree from each parent to its children.
- **Tree** messages, which are originated by the streamer when a request message is received. They are forwarded from parent to (one single) child, descending the tree until the downloader is added as a leaf somewhere.
- **Ping / Pong** messages, which are sent to the neighbors every n steps, in order to make sure that the neighbors are still alive.

Several important parameters can influence the outcome of the simulation, such as (i) the dissemination protocol (and relative parameters) employed to disseminate the request messages, (ii) the temporariness of the network (churns have a particular impact when the nodes leaving the network are involved in the streaming tree, thus leading to tree reconstructions), and (iii) nodes degree (lower is the maximum number of children of the nodes in the streaming tree higher is the height of the three and thus higher is the average latency as well).

4.3 Dissemination Protocol Variation

The choice of the employed dissemination protocol may have a significant impact on the network traffic. In the tested scheme, the request messages are the only ones spread into the network by means of a gossip algorithm, while the chunks are spread through the streaming tree. It follows that the number of forwarded items is proportional to the number of chunks to be sent, while the volume of the forwarded request messages is proportional to both the number of downloaders and the overall active participants in the system. Figure 1 shows the outcome of the metrics depending on the average number of neighbors per node, comparing the Fixed Probability and the Probabilistic Broadcast dissemination protocols. When the average number of connections per node and the forwarding parameter of the gossip algorithm are high, coverage and delay improve, but at the expense of more network traffic. Moreover, we have also previously demonstrated that the Degree Dependent algorithms have the best performances, but getting and updating the information about the degree of the neighbors might represent a significant cost that needs to be considered. Thus, other

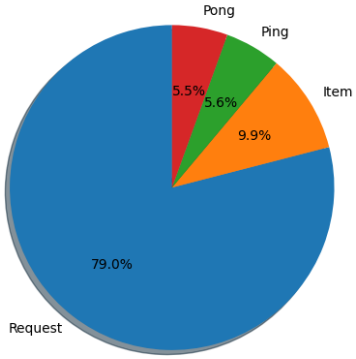
alternatives could be preferable, such as Fixed Probability, which by the experiments turns out to behave slightly better than Probabilistic Broadcast and just slightly worse than Degree Dependent algorithms.
Now, let us estimate the volume of network traffic for the estimated protocol. The algorithm used is Fixed Probability with 0.6 as the forwarding parameter (see Section 2). The chunk size in a P2P application typically varies between some hundreds of KBs and 1 MB (Song et al. 2020), and we assume that for our experiments all the chunks are 1Mb large. Excluding chunks, all the other messages carry very little information, so we can assume that their size in terms of bytes is equivalent to the minimum Ethernet frame size, which is 72 bytes (also considering the 8 bytes for the preamble) (Thiele and Ernst 2016), while the 12 bytes for the interframe gap are not necessary in this case, since all the small messages (i.e. the non item messages) are sent atomically. Let us consider a first basic case, where the deactivation rate for the node is 0% and thus the network is not temporal. Under this assumption, desirable but unrealistic in a real world scenario, no ping-pong messages are needed for coordinating the nodes, since participants cannot fail or leave during the execution. In a 2000 simulated entities scenario, using Fixed Probability with 0.6 as forwarding parameter (i.e. the minimum in order to achieve a full coverage), the network traffic is exactly split between chunks and the overhead. However, when simulating 10000 nodes, the impact of the Request messages increases in a relevant way, reaching up almost 90% of the network traffic. With the introduction of a significant deactivation rate (that is 0.1% for these experiments), the proportion in terms of messages weight is changing again as shown in Figure 1a. In this configuration, a ping-pong of messages occurs every 20 time-steps. It is possible to reduce the ping-pong frequency, with the aim to reduce the network traffic, but with the cost of worsening the chunks reception rate. In fact, undetected failures of nodes can cause important problems for both the overlay and the streaming tree. Of course, the higher the churn rate, the more important is to detect the failures as soon as possible. We can therefore suggest calibrating the ping-pong frequency based on the deactivation rate of the nodes.

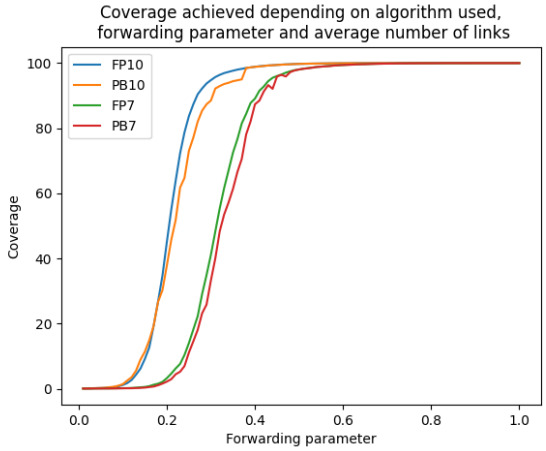
4.4 Network Temporariness

The deactivation rate has a big influence on the outcome of the tests, since the failures/exits of the non-leaf downloaders lead to a tree reconstruction, thus implying other downloaders to lose the stream for a certain amount of time (i.e. the time needed to detect the failure and join the tree again). The lower is the degree (for both the overlays), the more important is to detect the disconnection of the peers as soon as possible, in order to ensure a sufficient number of links for the nodes to both propagate the request messages through the mesh overlay and to spread the chunks efficiently in the streaming tree. In fact, forwarding chunks to a deactivated node is a waste of bandwidth, and furthermore undetected exits can also entail a drop of the degree of the streaming tree, thus worsening the efficiency of the propagation. Downloaders can assume their parent node has exited the network after a certain time if they do not receive messages anymore. On the other hand, detecting the left of the children (or the common neighbors is the mesh overlay) requires an active strategy. One idea is to employ a ping-pong of messages between neighbors at a certain frequency to ensure that participants are still alive and online.

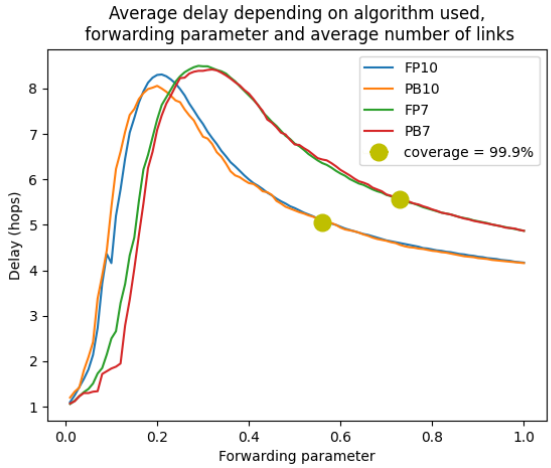
4.5 Tree Structure

In the proposed dissemination scheme, the delay for chunks delivery is proportional to the height of the streaming tree (or better, to the average depth of the downloaders), since on average more hops will be necessary to deliver a chunk. It follows that the maximum degree of the nodes is an important parameter, which can have a big influence on the outcomes. In a real-world scenario, the number of children for each downloader would depend on its upload bandwidth, and it would be convenient to put the nodes with more bandwidth availability at the top of the tree, in order to minimize the height. However, in our tests, we assume that all the nodes have a similar bandwidth availability, and thus we set a parameter, say d, to





(a) Less than 10% of the network traffic is actually used for sending the requested items.



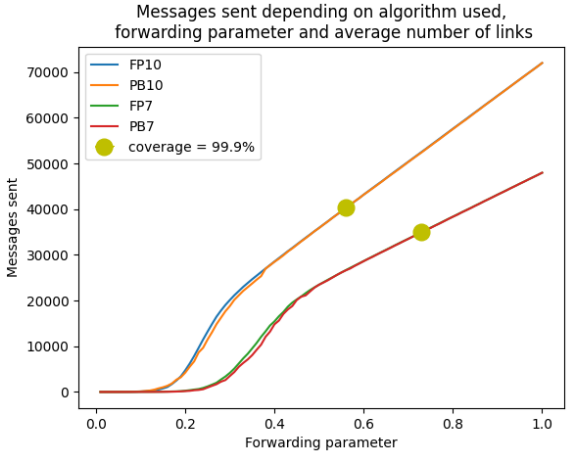
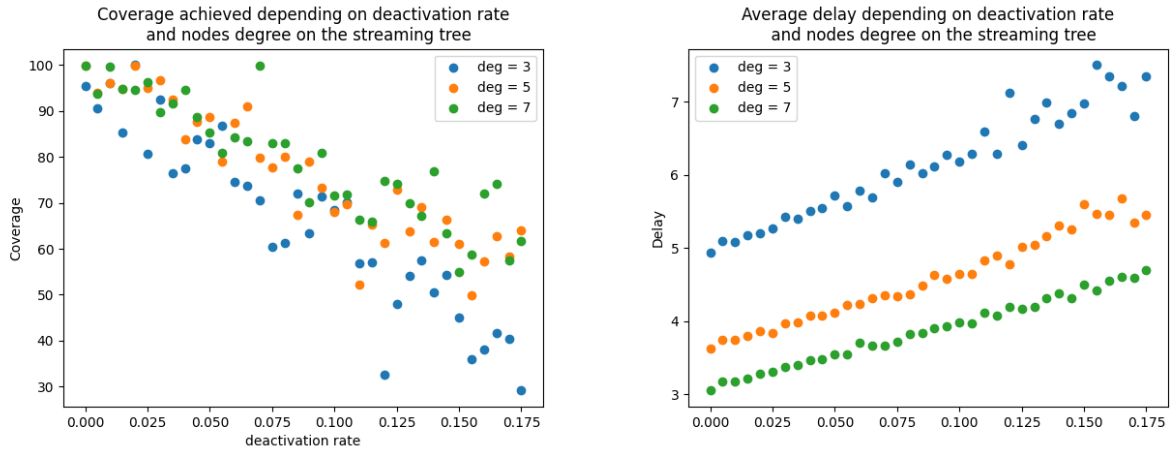


Figure 1: The marker indicates the minimum forwarding value to achieve a coverage of 99.9%. FPx = Fixed Probability having x as the average degree. PBx = Probabilistic Broadcast having x as the average degree.

indicate the nodes degree. A strategy for minimizing the tree height would be to adopt a balanced tree, however, the overhead to manage the balance of the tree could exceed the benefits.
Figures 2a and 2b show the interdependence between d and both delay and chunk delivery rate. When the parameter d is high, the average depth of the downloaders in the streaming tree decreases, and a greater number of nodes in the streaming tree are leaves. This leads to fewer tree reconstruction events since the amount of requests for re-joining the streaming tree when a failure occurs is proportional to the number of descendants of the node that failed. On the other hand, the delay is inversely proportional to the average height of the steaming tree, thus d is another relevant factor.

4.6 Preferential Attachment

A specific extension of the LUNES-Temporal simulator was developed for evaluating also a preferential attachment strategy. The attachment of the nodes was the only aspect of the simulator where the simulated entities were exploiting a priori knowledge, assuming to know some active nodes to connect with, which



(a) (b)

Figure 2: Average delay and chunk reception rate worsen as the deactivation rate increases.

were chosen at random among the set of active nodes. In this extension of LUNES-Temporal, instead, the attachment is managed by means of supernodes, which are special nodes (100 in our tests out of 10000 simulated entities) whose address is publicly known and that we assume to be always active. By contacting a pair of supernodes, the new nodes entering into the system receive the contacts of a list of peers to establish connections with. We then introduced a new field in the structure of the simulated entities, called stability coefficient, which provides all the peers a different probability to deactivate. In the previous version, in fact, the probability for the nodes to activate and deactivate at a certain time-step was homogeneous for all peers. Differentiating such an aspect allows us to "reward" the most stable nodes, whose contact is given more often to the entering peers. In fact, selecting preferentially the most stable nodes should lead to a more stable network, with consequent positive effects on the performances. The supernodes are not aware of the stability coefficient of the nodes, thus they evaluate the stability based on the number of time-steps a peer is active, assuming that statistically the more time a node is connected the more probable it is to deal with a stable peer. Figure 3 shows that slight differences in degree distribution occur between the two approaches. In both cases, with an average of 10 connections per node (supernodes excluded), most of the nodes tend to have a degree between 5 and 12 but with a preferential attachment strategy, there are multiple nodes having a considerably larger degree than the average, because of their stability leading to be preferred as a neighbor.

5 SIMULATOR PERFORMANCE EVALUATION

In this section, a performance analysis of the LUNES-Temporal simulator is reported. As already mentioned, we in fact need to understand if the simulator is able to scale, based on the size of the simulated system, the type of simulated scenarios, and the level of dynamicity. To this extent, we measure the time to complete the simulation when varying i) the number of simulated nodes (the higher the number the higher the complexity); ii) type of attachment for the creation of the P2P overlay; iii) probability of forwarding the messages (that influences the number of generated messages and thus the complexity of the model); iv) deactivation rate (that varies the dynamism of the temporal graph simulating the P2P system).
The tests have been executed on a PC equipped with an Intel Core i5 processor (10th generation) with 16 GBs of RAM running GNU/Linux Fedora 35 (kernel Linux 5.11.0-41-generic).
As expected, the execution of the experiments is time-dependent on the identified parameters, in particular on the number of nodes. LUNES-Temporal is proved to manage efficiently simulations with up to 10000 nodes, but the time required for the tests grows exponentially with the increase of the agents, thus for carry-

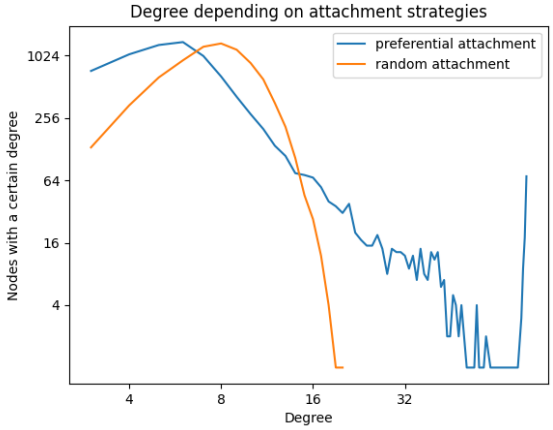
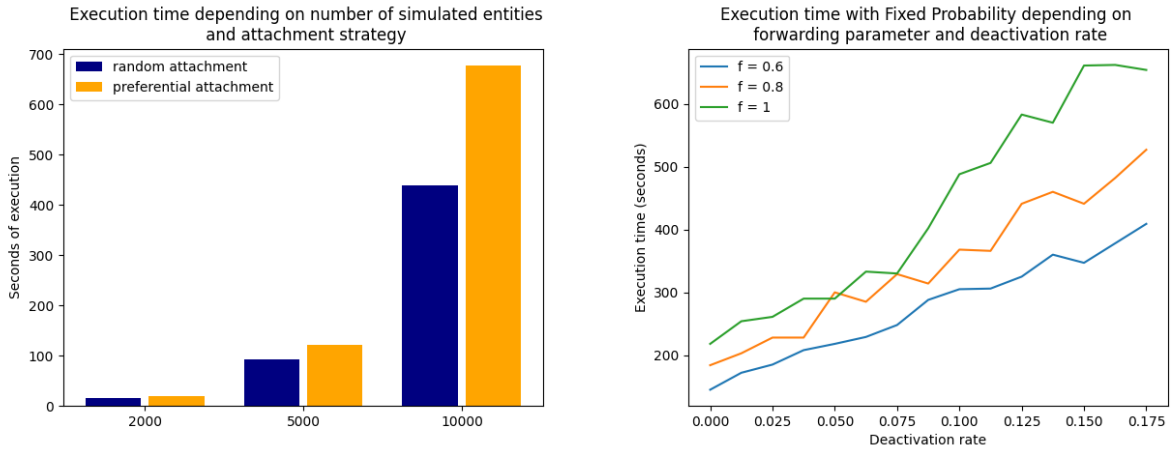


Figure 3: Comparison of degree distribution between preferential and random attachment. Log-log scale.

ing out experiments with a significantly greater number of nodes other simulation techniques might be used, such as employing a parallel and distributed approach. Figure 4a displays the running times (with 0.1% as a deactivation rate and pure broadcast as dissemination protocol) with a varying number of nodes, showing that the time needed to complete 20 epochs of the simulation grows exponentially with the increase of the simulated entities. Furthermore, it is worth noticing that the preferential attachment strategy mentioned in the above section requires more computational effort, above all with a large number of nodes involved in the simulation. In any case, given the limited computational capacity of the host where the simulations have been executed, these tests allow concluding that LUNES-Temporal is able to easily handle large-scale simulations of temporal graphs.

On the other hand, also the temporariness of the network has an impact on the overall running time. Figure 4b shows how the execution times change by varying the deactivation rate, growing constantly as such a parameter increases. These results confirm that the level of dynamism has a strong impact on the complexity of the simulated models and suggest that effective simulation tools need to be designed for supporting the study of temporal network based models. In this sense, however, LUNES-Temporal seems to be an adequate and viable tool for the execution of the considered simulations.



(a) (b)

Figure 4: Time to complete the executions of 20 epochs

6 CONCLUSIONS

In this work, we described a simulator that we developed for the study of complex systems modeled through temporal graphs. To this aim, we simulated a complex peer-to-peer environment, where participants can join a data stream by relying on dissemination protocols, in a totally decentralized way. Due to the dynamicity of the system, characterized by frequent arrivals and departures of nodes, a temporal graph has been used to model the overlay network, allowing us to represent churns, which are one critical element for these scenarios. LUNES-Temporal, the simulator employed for the performance evaluation, easily enables the modelling of the peer-to-peer system and its algorithms, even without going into details for what concerns certain low-level aspects, such as Internet protocols and the geographical position of the nodes. On the other hand, the tests have been focused on the comparison among different strategies applied on the overlay network, in order to gain information about what are the most important choices for appropriate functioning of the protocol and which solutions provide the better benefits. We believe that this approach is suitable for the simulation of such scenarios since it can provide significant results with a relatively limited computational effort. By employing a lightweight simulation tool that directly focuses on the most important issues concerning the functioning of the system, we were able to perform several tests with a large number of simulated entities in a limited amount of time.
Regarding the outcome of the experiments, we proved that organizing the downloaders into a tree can significantly help the propagation of the chunks, but also the choice of the gossip algorithm and its parameters is very important in the attempt to minimize the network traffic. Besides that, also factors like the degree of the nodes in the overlays and the frequency of the ping-pong messages have relevant effects on the global metrics, but the most impactful element on the outcome is undoubtedly the level of the temporariness of the network, both for which concerns the chunk reception rate achieved and the performance of the simulation. In fact, the exit of a node might trigger a sequence of interactions based on messages among several simulated entities, resulting in more computational effort. The time required to complete a simulation increases with the number of participants, but from our tests, it turns out that LUNES-Temporal can easily manage several thousands of simulated entities.

ACKNOWLEDGMENTS

This work has received funding from the EU H2020 research and innovation programme under the MSCA ITN European Joint Doctorate grant agreement No 814177, LAST-JD-RIoE.

REFERENCES

2022. "LUNES-Temporal". https://github.com/luca-Serena/LUNES-temporal2/.

Antulov-Fantulin, N., A. Lančić, T. Šmuc, H. Štefančić, and M. Šikić. 2015. "Identification of patient zero in static and temporal networks: Robustness and limitations". *Physical review letters* vol. 114 (24).

Cigno, R. L., A. Russo, and D. Carra. 2008. "On some fundamental properties of p2p push/pull protocols". In *2008 Second International Conference on communications and electronics*, pp. 67–73. IEEE.

Cohen, B. 2003. "Incentives build robustness in BitTorrent". In *Workshop on Economics of Peer-to-Peer systems*, Volume 6, pp. 68–72. Berkeley, CA, USA.

D'Angelo, G. 2017. "The Simulation Model Partitioning Problem: an Adaptive Solution Based on Self-Clustering". *Simulation Modelling Practice and Theory (SIMPAT)* vol. 70, pp. 1 – 20.

D'Angelo, G., and S. Ferretti. 2017. "Highly Intensive Data Dissemination In Complex Networks". *Journal of Parallel and Distributed Computing* vol. 99, pp. 28–50.

Hiraoka, T., N. Masuda, A. Li, and H.-H. Jo. 2020. "Modeling temporal networks with bursty activity patterns of nodes and links". *Physical Review Research* vol. 2 (2), pp. 023073.

Magharei, N., R. Rejaie, and Y. Guo. 2007. "Mesh or multiple-tree: A comparative study of live p2p streaming approaches". In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, pp. 1424–1432. IEEE.

Malatras, A. 2015. "State-of-the-art survey on P2P overlay networks in pervasive computing environments". *Journal of Network and Computer Applications* vol. 55, pp. 1–23.

Neysiani, B. S., N. Neematbakhsh, B. Barekatain, M. A. Maarof, and K. Zamanifar. 2012. "Understanding pull-based method efficiency in peer-to-peer live video streaming over mesh networks". *Journal of Basic and Applied Scientific Research* vol. 2 (11), pp. 11626–11643.

Ouali, A., B. Kerherve, and B. Jaumard. 2009. "Toward improving scheduling strategies in pull-based live P2P streaming systems". In *2009 6th IEEE Consumer Communications and Networking Conference*, pp. 1–5. IEEE.

Pianese, F., D. Perino, J. Keller, and E. W. Biersack. 2007. "PULSE: an adaptive, incentive-based, unstructured P2P live streaming system". *IEEE Transactions on Multimedia* vol. 9 (8), pp. 1645–1660.

Riad, M. H., M. Sekamatte, F. Ocom, I. Makumbi, and C. M. Scoglio. 2019. "Risk assessment of Ebola virus disease spreading in Uganda using a two-layer temporal network". *Scientific reports*.

Serena, L., M. Zichichi, G. D'Angelo, and S. Ferretti. 2021. "Simulation of dissemination strategies on temporal networks". In *2021 Annual Modeling and Simulation Conference (ANNSIM)*, pp. 1–12. IEEE.

Song, Y., H. Ni, and X. Zhu. 2020. "Analytical Modeling of Optimal Chunk Size for Efficient Transmission in Information-centric Networking".

Thiele, D., and R. Ernst. 2016. "Formal worst-case performance analysis of time-sensitive ethernet with frame preemption". In *2016 IEEE 21st International Conference on Emerging Technologies and Factory Automation (ETFA)*, pp. 1–9. IEEE.

Yao, Z. 2009. *Understanding churn in decentralized peer-to-peer networks*. Texas A&M University.

Zhang, M., Q. Zhang, L. Sun, and S. Yang. 2007. "Understanding the power of pull-based streaming protocol: Can we do better?". *IEEE Journal on Selected Areas in Communications* vol. 25 (9), pp. 1678–1694.

LUCA SERENA is a doctoral researcher in Computer Science at University of Bologna. His research interests include simulation, distributed systems, computer security and blockchain technologies. His email address is luca.serena2@unibo.it.

MIRKO ZICHICHI is a doctoral researcher in the Law, Science and Technology Joint Doctorate - Rights of Internet of Everything, funded by Marie Skłodowska-Curie Actions. His doctoral research focuses on the use of Distributed Ledger Technologies and Smart Contracts for the protection and distribution of individuals' personal data. His email address is mirko.zichichi@upm.es.

GABRIELE D'ANGELO is an Assistant Professor at the Department of Computer Science and Engineering, University of Bologna. His research interests include parallel and distributed simulation, distributed systems, online gaming and computer security. Since 2011 he has been in the editorial board of the Simulation Modelling Practice and Theory (SIMPAT) journal published by Elsevier. His email address is g.dangelo@unibo.it.

STEFANO FERRETTI is an Associate Professor at the Department of Pure and Applied Sciences, University of Urbino "Carlo Bo", since 2020. His research interests include distributed systems, complex networks, data science, fintech and blockchain technologies, multimedia communications, hybrid and distributed simulation. His email address is stefano.ferretti@uniurb.it.