Simulating Search Protocols in Large-Scale Dynamic Networks

Spiridoula V. Margariti
Department of Computer Engineering
T.E.I. of Epirus, Arta, Greece, GR-47100
and
Department of Computer Science and Engineering
University of Ioannina, Greece, GR-45110
Email: smargari@cse.uoi.gr

Vassilios V. Dimakopoulos
Department of Computer Science and Engineering
University of Ioannina, Greece, GR-45110
Email:dimako@cse.uoi.gr

Abstract

Reproducing complex networks with features of real-life networks is exciting and challenging at the same time. Based on the popular Omnet++ discrete event simulator, we introduce Armonia, a framework for modeling massive networks and their dynamic interactions. It includes a collection of topology generators, a set of resource placement and replication modules, a component for specifying resource location strategies, while also offering support for exporting data in order to visualize or analyze with other appropriate tools. Our framework targets search protocols in large-scale dynamic networks. Here, we apply it to simulate various probabilistic flooding strategies, making a comparative study of their performance over different network topologies.

1 Introduction

Complex systems, such as electrical power grids and telephony networks, social relations, the World-Wide Web and the Internet, collaboration and citation networks of scientists, are large and scale free [22]. They represent a set of entities, e.g. individuals, defined in an abstract space (the nodes) and the relations among them (the links) [5] that are dynamically evolving in time. To deal with the complexity of these networks, researchers turn to graph theory and, particularly, random graphs.

Modeling and studying complex systems / networks purely analytically is not always a viable approach; there is a need for a simulation framework in order to gain insights to their behavior, to test and evaluate new ideas, to reproduce results, to compare with empirical results or to validate other research. On the other hand, it may be simply impossible to access and experiment with a real system, while a simulation framework provides all the necessary facilities to interact with such a system. Studies rely on simulation in order to understand the structure, formation, function and any ongoing processes in the actual system.

A realistic simulation framework is essential for at least three reasons. The first is that it can reproduce real systems and their interactions, for example, how nodes are connected or what is the nodes' importance according to their position in network. The second is that it can be employed to design, implement, evaluate and propose new ideas or protocols [3]. The last reason is that it can be used to confirm analytical results, to gain useful insights or to reveal hidden properties.

Even though these systems may have different attributes (e.g. channel characteristics, mobility features) they can be represented as a network graph. The simulator operates in abstracting fashion, provides a tunable environment via module and parameters and removes the complexity of construction of the network topology.

Given the network, searching is one of the fundamental issues. It refers to the ability to discover and locate a target (a person in a social network, a resource in p2p network, a router in a communication network) without complete knowledge about the network topology and/or the target placement [6]. To achieve this, the node uses only local information and applies a search policy to forward the request to some of its neighbors, which in turn push it to their neighbors, and so on, in order to reach the target. The problem of searching has received a lot of attention by the research community. A significant number of algorithms have been proposed, analyzed and compared experimentally using a multitude of custom simulators, which in most cases are used only to support one particular proposal ([18], [19]).

In this work, we introduce a novel network simulator, called *Armonia*. To the best of our knowledge this is the first general-purpose simulator tool which allows the simulation of arbitrary search protocols over arbitrary network topologies. We present the design of a framework that can facilitate the reproduction of realistic networks, exhibiting churn and dynamicity, while evolving over time. Armonia is a flexible tool that combines a set of components to perform three main functions: modeling (arbitrary topologies), simulating (search and replication strategies) and analyzing (collecting statistic results). To deploy these features, the users may customize the framework according to their interests, with respect to overlay topology (which can be static or dynamic), resource placement and search strategy. There are a number of predefined modules for the network topology, the search protocol and resource allocation, but the framework is extensible, allowing the implementation of additional models and policies as module plug-ins. Here is a summary of our contributions:

- We present a novel, general-purpose simulator for simulating realistic networks, exhibiting churn and dynamicity.

- We provide a flexible framework to investigate the behavior of various search protocols and replication strategies in complex networks.
- We examine the performance of probabilistic flooding protocols in static and dynamic networks.

The remainder of this paper is organized as follows. Section 2 discusses related work and surveys relevant tools. Section 3 presents the Armonia simulation framework, and gives a detailed description of its architecture. Main findings, simulation results and discussions are reported in Section 4. Finally, Section 5 concludes the paper.

2 Related Work

Significant effort has been directed towards implementing a vast amount of simulation tools for complex networked systems. We distinguish these simulator classes as follows:

(a. short-lived, custom ones, which constitute the most common type of simulators for personal use [18], [19]. They are usually written by authors to support their works,

(b. detailed, low-level simulators, which are general purpose network simulators focusing on the physical network (such as the well-known NS-2, NS-3). They provide substantial support for simulating, transportation media, devices and networking protocols, and

(c. application-level ones, that are also general purpose simulators, aiming at large scale overlay networks such as p2p networks, social networks, biological networks.

We are interested in the last class, and we strive to implement a framework that can facilitate the reproduction of realistic networks, exhibiting churn and dynamicity while evolving over time. The bibliography, on the networks we are interested in, bases the experimental results mostly on static snapshots of the systems, while our pursuit is to study environments where nodes join and leave continuously.

Short-lived simulators are bespoke and very difficult to compare with one another or to extend. In addition, they also involve significant duplication of effort.

Regarding low-level simulators, NS-2 and NS-3 are among the most popular. NS-2 [16] is a discrete event simulator written in C++ and OTcl. NS-2, and its successor NS-3, support research and education on networks and internet systems. NS-3 provides models of how packet data networks work and perform, and offers a detailed engine for users to conduct simulation experiments.

Application-level frameworks can be used to experiment with possibly complex and varying conditions, in order to address a sufficiently wide range of interactions on a network. These simulators are composed of two major parts: a) the nodes and b) the network connecting all nodes. A *node* incorporates resources (e.g. files, memory, energy). The *network* is often very large (connecting millions of nodes), static or dynamic where the interactions between the nodes are rapidly changing and evolving. The network is used to model in detail the structure of real systems. PeerSim [17], SNAP [12], Omnet++ [23], and Oversim [4] belong to this class of simulators.

PeerSim [17] is a Java-based p2p system simulator with two simulation engines: a cycle-based one and an event driven one. It offers predefined models such as Pastry, Chord, Kademlia, Skpnet, Bittorrent, TMan and Cloudcast while the user can build its own protocol. The cycle-based engine is scalable as it avoids the overhead required to simulate low level communication but it provides for less realistic models than the event-based engine.

The Stanford Network Analysis Platform (SNAP) [12] includes a general purpose, high performance system for analysis and manipulation of large networks. The core SNAP library is written in C++/Python and optimized for maximum performance and compact graph representation. It easily scales to massive networks with hundreds of millions of nodes, and billions of edges. It efficiently handles large regular and random graphs, calculates structural properties, and supports attributes on nodes and edges. It is, however, designed to simulate only the evolution of the network topology.

Omnet++ [23] is an open source simulation library and framework, widely used for the simulation of communication protocols and networks. It is modular, component-based and written in C++. Due to its generic and flexible architecture it is primarily used for building network simulators in various domains such as communication networks, complex systems, queuing networks or hardware architectures. Our simulation framework is based on Omnet++.

Oversim [4] is a C++-based open-source framework for p2p networks designed to work over the OMNet++ simulation environment. The simulator offers several overlay models for structured (e.g. Chord, Kademlia, Pastry) and unstructured (e.g. GIA) p2p systems and protocols. The user can extent the model with additional protocols or network topologies.

SNAP supports only non-dynamic networks, i.e. it does not cover the addition or removal of nodes after the formation of the topology. On the other hand, Oversim requires significant effort to accurately implement a network model. In PeerSim the network is completely abstracted, precluding studies that would mandate different implementations of the underlay networks, including realistic attributes (e.g. latency). Another drawback of PeerSim is the fact that it has no support for gathering statistics. PeerSim and Oversim were specifically designed for simulating p2p networks while Omnet++ is a general purpose simulator. Moreover, Omnet++ covers the need for precise network simulation. Based on the above, we consider Omnet++ as the most appropriate basis for our purposes here.

Another flexible infrastructure for simulating large-scale complex networks is the DRGSimLib toolbox [10] which is also implemented on top of Omnet++. It operates in a high abstraction level, and the interactions between simulated objects mimics the real (communication) networks. The networked system is represented as a graph and its topology is implemented by a generator plug-in. We will give more details about this library because, in part, the topology component of our framework was inspired by it.

The main components of the toolbox are the *node*, the *nodeFactory*, and the *topology*. The node is a compound module which is used to represent each simulated object, as shown in Fig. 1. The nodeFactory component manages the creation and deletion of nodes while the topology component handles the nodes relations or network connections.

The nodeFactory component operates independently of the other components. It manages the population of nodes in the system. It generates and deletes nodes according to their lifetime specifica-

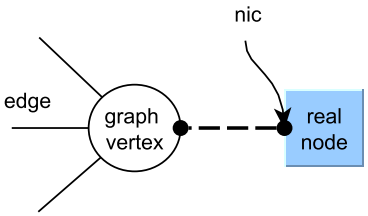

Figure 1. A network node corresponds to a graph vertex

tions. The node module includes an essential part, the *topologyControlNic* (nic) that registers or unregisters a node with the topology. The nic acts as an interface between a simulated object and a graph vertex. It is aware of which are the neighbors and of any changes in the local neighborhood.

The topology component manages the generation and the maintenance of the graph which represents the networked system. It consists of a *topologyControl* and a *generator* module. The generator acts as a plug-in and handles the network formation. The tpologyControl oversees the graph and calls the generator to add or remove nodes, and in conjunction with the nic, maintains the graph structure. The topology component is responsible for the close correspondence of each graph vertex to each node and for handling nodes churn.

Summarizing, the available application-level simulator packages provide models only for the network topology and its evolution. They serve the purpose of studying how the structure of the system behaves over time and to derive mostly graph-theoretic metrics. There are no known general-purpose simulators that allow experimentation with search and replication algorithms, which can be effected on different network topologies, even though these problems have received considerable attention in the open literature. Our simulator serves to fill the gap, and is presented next.

3 Our Simulation Framework: Armonia

We envisage a simulator framework that makes it easy to study arbitraty search and replication strategies in arbitrary complex dynamic networks. While the other known simulators aim to model large-scale networks, we seek to model complex interactions over such networks, such as searching or resource allocation. Our simulator is multi-purpose and consists of several components:

- The *topology* component, which consists of nodes and their links, and can be used to define arbitrary structures. A number of distinct, ready-to-use topologies are offered.
- The *resource and replication* component, a collection of modules to control the creation and the allocation of resources.
- The *search protocol* component, which is responsible for applying a predefined search method on the overlay network.
- The *statistics* component, which is used to collect global or local statistics according to user requirements.

Fig. 2 shows an example of a simple network and its structure. It includes the nodes, as presented by the Omnet++ graphical interface, along with the topology as visualized using the well-known Gephi package [2].

3.1 The Topology

As mentioned above, the DRGSimLib library [10] formed the basis for the topological part of Armonia. We have modified that library considerably, with new functional components and structural enhancements. We have also introduced additional network topologies. In particular, we maintain the DRGSimLib abstraction for the network, which consists of a topology management component (*topologyControl*) and a collection of model networks (*generator*) with some adaptations. In particular:

- New network models have been added.
- Multiple edges between a pair of nodes can be disallowed.
- The clustering property has been added to network model.
- The node component is redesigned and new features are added.

Taken together, these modifications add more capabilities and functionalities to the simulator.

New network models The topologies implemented by the authors in [10] include Barabási, FullyConnected, random, Tree and gnpRandom (a modified version of Erdős-Rényi netowrk). We have implemented the classical Erdős-Rényi random graph [8] and the Caveman-Solaria model, which enables the formation of networks with the small world property [24]. Moreover we implemented a new class that provides the ability to access and proccess other large scale network datasets (such as the ones included with SNAP [12]).

Multiple edges Based on user's choice, the connections among a node pair (u, v) can be limited to at most one edge. This way, we avoid setting multiple edges between a pair of nodes at network creation or network evolution, which might affect the correct operation of certain search protocols.

Clustering coefficient Clustering is a common characteristic found in many real networks and it expresses the property that two neighbors of a node v may also be neighbors themselves [20]. The clustering coefficient is given by:

$$cc = \frac{3 \times N_{\Delta}}{\sum_v \binom{d_v}{2}}$$

where d_v is the degree of a node v and N_{Δ} is the number of triangles (i.e. triads of nodes which are neighbors to each other) present in a network. The clustering property is a crucial factor [1] and affects the functionality and the structure of the network. The user adjusts it via a configuration parameter (*clusterCoefficient*) which has been added to the topology generator. Setting it to zero effectively disables this property.

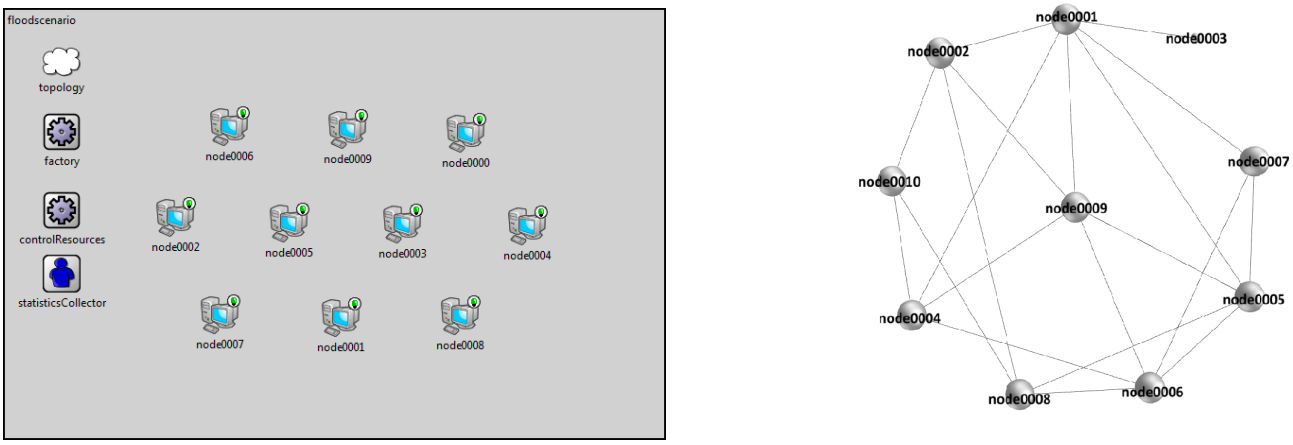

Figure 2. An instance of a simple Armonia simulation: the Omnet++ network (left) and the corresponding topology (right).

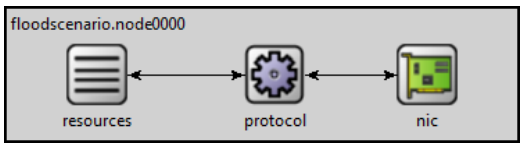

Figure 3. The structure of each node

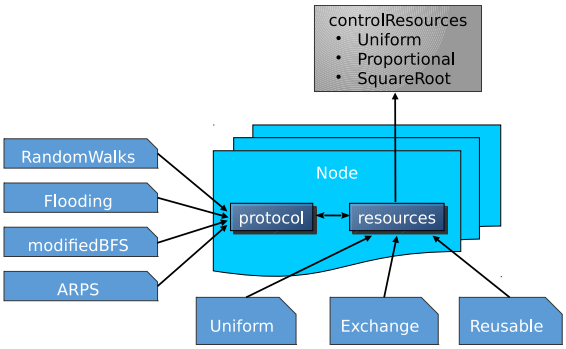

Figure 4. The use of the protocol and resource modules in Armonia.

The node component The node represents an entity along with its attributes. The component maintains the original communication module (*nic*). In addition, a protocol and a resource interface are in effect, as depicted in Fig. 3. The operation of the protocol module is specialized to satisfy the requirements imposed by the search strategy and the replication policy. The resource interface is used to manage the resources offered by the node. Protocols and resources, which inherit from respective base classes, can be specified in a configuration file.

3.2 Resource and Replication

Resources are distributed across the network at various nodes and can be accessed by other nodes. Important issues in such an environment include:

- The *resource distribution* model, which defines how many distinct resources are available in the system and how they are assigned to nodes.
- The *replica distribution* model which specifies how many replicas of each resource are present in the system.
- The *resource acquisition* model, which focuses on obtaining the right resources that meet each node needs.

Our resource management framework is quite general, allowing researchers to plug in their own resource management policy.

The *controlResource* module is a simple Omnet++ module and is used to handle the resource and the replica distributions. Assuming that there are m distinct types of resources on the system and each resource i is replicated on r_i nodes, the total number of resources is $R = \sum_{i=1}^{m} r_i$. We implement three static replication distribution schemes, namely uniform, proportional and square-root [14], as follows:

- *Uniform*: all resources are replicated to the same number of nodes ($r_i = R/m$)
- *Proportional*: the replication of a resource i is proportional to its relative popularity, denoted as q_i and defined as the portion of queries issued for the i^{th} resource ($r_i \propto q_i$).
- *Square-root*: the replication of a resource i is proportional to the square root of its relative popularity ($r_i \propto \sqrt{q_i}$).

The parameters of the controlResource module include the maximum number of different resources in the system (*maxNumOfResources*) and the name of the replica distribution scheme (*distributionName*).

Each node is assigned and maintains a number of resources. In addition to resource and replica distribution, the node's *resource* module handles resource acquisition. At node initialization, the resource module acquires resources from the controlResources module using the *uniform* allocation policy, whereby resources are chosen uniformly randomly from the global set of all available resources.

After the formation of the network, when churn commences and nodes start entering and leaving the network, each joining node acquires resources according to one of three predefined policies: uniform, reusable and exchange. In the uniform policy, resources continue to be chosen uniformly randomly from the global set of all available resources. In the *reusable* policy, before a node leaves the system, it returns its resources to the controlResources module for reuse. Then, when a new node joins the system, the controlResources module will assign from the reusable resources, if they exist. Finally, if the *exchange* policy is in effect, a node asks its neighbors for resources, selects a random set and replicates them locally. The above policies allow us to model behaviors that occur in real systems. For example, the exchange policy is used in p2p systems (e.g. in Gia [7]) and virtual communities, while the reusable policy can be used to model the resource management scheme in Hadoop [25].

3.3 The Search Protocol

Searching is a fundamental process and is used to discover services and resources, to locate cooperators and to discover trust networks [13]. In Armonia, the *search* component is responsible for applying the search algorithms over the network. It plugs into the node component via the protocol interface, which inherits from the generic *IProtocol* interface. Its objective is twofold: first, it facilitates the utilization of diverse search algorithms, simply by specifying a protocol class name in the simulator configuration file, and second, it carries out all the common tasks required by the search process.

Our infrastructure includes a mechanism which can help avoid the further forwarding of an already sent message. In effect it implements a local monitor at each node connection in order to limit retransmissions on that edge. It also provides a handler for resource allocations during the initialization phase of new nodes (e.g. when the exchange resource acquisition policy is in effect). To let a node know if a given query was received and transmitted in the past, we maintain a *pastQueriesBuffer* structure that stores past queries. Users can limit the capacity of pastQueriesBuffer via the *pastQueryCounter* configuration parameter. Moreover, we mark the edge where a message came from, and (based on a configurable flag) we do not include it in the forwarding group.

The above mechanisms can be used by any search policy and are optionally bypassed, if required. The search procedure is completed by considering the initialization of a query and the forwarding group selection. The whole infrastructure simplifies considerably the implementation of new search schemes. We currently offer a number of search strategies:

- *Flooding*, which is probably the most well-known and generic strategy. It is frequently used in the absence of any information about resource placement. In (plain) flooding, the node that initiates the search sends a query message to all its neighbors. Any neighbor that does not know about the resource propagates the message to all its neighbors and so on, until the resource is discovered or some termination conditions are reached. Flooding-based strategies commonly use a Time-To-Live (TTL) parameter, in order to limit the forwarding of the query message beyond a predefined number of hops (steps). Flooding is generally fast, but produces an excessive number of messages, many of which are redundant.
- *Random walkers* [14, 9]. In the standard random walker, when a node receives a query and does not hold the desired resource, it forwards the message to one, randomly chosen neighbor. Alternatively, one can employ multiple walkers to propagate the query in parallel and locate the resource faster, at the expense of more message transmissions.
- *Modified BFS* [11], where a query message is forwarded to each neighbor with a fixed probability (called *forwarding probability*), in order to reduce the message overheads of plain flooding.
- *ARPS* [26], which utilizes different forwarding probabilities for different resources, depending on estimations of resource popularity.
- *APF* [15], where the forwarding probability varies according to the distance from the query initiator and utilizes knowledge about resource popularity.

3.4 Statistics

The collection of statistics serves two purposes; first, the user can obtain measurements about the system as a whole (global statistics) and, second, a node needs to collect results to estimate some network or resource metrics, such as relative resource popularities, which may be useful for the interactions with other nodes (local statistics).

The component *statisticCollector* is a simple Omnet++ module and is used to handle aggregate statistics from the entire network. Statistics have the following format:

$$\langle \text{statName}, \text{variable1}, \text{variable2} \rangle,$$

where *statName* is the name of the metric (e.g. messages) and *variable2* aggregates and describes measurements when *variable1* has a certain value; for example, if statName is used to count messages, and variable1 stores the time-to-live (TTL) value, variable2 counts the total number of messages for this particular TTL.

For the collection of local statistics, we constructed a *localStatistics* class, which is included in the node component. It uses the same format as the global statistics module.

4 Simulation Study and Discussion

For a search performance simulation study, one has only to define (via configuration parameters) the desirable search policy, the network structure and the replication strategy. When the simulation sessions finish, the globally collected statistics (statisticCollector module) are used to assess the search performance.

A sample simulation study is presented here and constitutes a comparison of the performance of various search strategies on different topologies. We measure the performance using three metrics:

- The *probability of success*, which measures the probability that a query can locate the desired resource, given a TTL hop limit. A query is considered successful if it discovers at least one replica of the resource in question.

Table 1. Simulation parameters for static networks

graph	parameter	nodes	r_i	flood	mBFS	ARPS
ER	$d = 6$	25000	30	$p_f = 1$	$p_f = 0.5$	$p_f = 0.8$
BA	$m = 2$	10000	15	$p_f = 1$	$p_f = 0.5$	$p_f = 0.8$

- The total number of *messages* that are transmitted, before the resource is located.
- The number of *duplicate messages*. A message is considered duplicate, or redundant, if it is received by the same node more than once (and thus does not contribute to the success of the search). It serves as a measure of the how efficiently a policy utilizes the network resources.

We compare four flooding-based searching policies, namely plain flooding (flood), modified BFS (mBFS), Adaptive Resource-based Probabilistic Search (ARPS) and APF. All strategies operate under a (variable) TTL hop limit. For the last three strategies, the forwarding probabilities (p_f) follow the corresponding authors' guidelines. In particular, in mBFS the query is forwarded to 50% of a node neighors ($p_f = 0.5$), while a value of $p_f = 0.8$ is used for ARPS. For plain flooding $p_f = 1$, effectively contacting all neighbors in every case. The forwarding probabilities of APF involve a different calculation in each node [15]. All the search policies utilize the pastQueriesBuffer mechanism, with a capacity of 100 queries, for reducing the number of redundant messages.

The two topologies we base our study on are the classical Erdős-Rényi (ER) random graphs and Barabási (BA) power-law graphs. The Erdős-Rényi network is formed on N nodes and each pair of node is connected by an edge with probabilty $p \in (0, 1)$. The expected average degree (d) is given by $d = pN$.

The Barabási network begins with m_0 connected nodes. Each newly inserted node gets connected to m existing nodes.

We initially consider static networks, which means that after the topology is formed, no other nodes enter or leave the network. We use the uniform allocation and distribution policy. A predefined number of replicas (r_i) for each resource are randomly uniformly placed on the nodes, where the value of r_i is same for all resources i. Each node may submit a query until a maximum total number is reached. For these of experiments the total number of queries was limited to 1000. The simulation parameters are summarized in Table 1.

The simulation results are shown in Fig. 5–6. The figures plot both the probability of success and the number of duplicate messages. For the particular configuration we have chosen, all search strategies are successful at locating the required resources, as long as TTL has a value of 3 or more. The mBFS strategy requires at least 5 steps, while ARPS needs at least 4 hops in the power-law network. The efficiency of the strategies shows up vividly in the number of duplicate messages. While mBFS is slower to discover the resources, it also produces less redundant traffic, in both networks. ARPS, on the other hand, seems to behave almost like plain flooding in the ER network; it seems slower but more efficient in the BA case. Notice that in the Barabási network (Fig. 6), we simulated only the three of the search policies, as APF requires some network metrics (in particular the average node degree at a given distance from the query initiator) that are not known analytically known in power-law networks. In the ER case, however, the APF protocol achieves the lowest number of duplicate messages.

Table 2. Simulation parameters for dynamic networks

churn distr.	ER ($d = 6, r_i = 25$)	BA ($m = 2, r_i = 15$)
ND	$\mu = 4$h, $\sigma = \mu/3$	$\mu = 4$h, $\sigma = \mu/3$
WD	$a = 4$h, $b = 0.5$	$a = 4$h, $b = 0.5$

We next consider dynamic networks. *Churn* is the continuous process of node arrival to and departure from the network [21]. A node participates in the system for a certain time period, called the node *lifetime*, which is the elapsed time from the first appearing in the system until its permanent departure. For modeling lifetime, any predefined Omnet++ distribution can be used. For this set of experiments, we chose two of the most commonly used lifetime distributions when studying complex networks: the Normal distribution (ND), with a mean (μ) and standard deviation (σ) parameters and the Weibull distribution (WD) with parameters a and b. For all of them we set the same mean ($\mu = 4$ hours). We consider Poisson arrivals. For the ER network with 5.000 initial nodes, the node arrival rate of $1/\lambda = 4$ sec was used, while the BA network started with 10.000 nodes, and used $1/\lambda = 2sec$. As in the static case, we use the uniform allocation and distribution policy and we simulated up to a total of 1000 submitted queries. The simulation parameters for the dynamic case are summarized in Table 2.

For the dynamic case we only consider the plain flooding search strategy here. The performance measurements are illustrated in Fig. 7–8. On the left side of the figures, we show the evolution of the network for the first 100 sec of simulation time, while churn is in effect, as a function of time t. For this specific scenario, the curves in both networks have approximately the same shape. For the first few hops the Weibull distribution results in a larger number of total messages than the Normal distribution; after that the opposite is observed, which can be probably explained by the fact that the particular Weibull distribution results in a large number of node removals.

Also, the curves highlight the differences among the two lifetime distributions during network evolution. For this particular search strategy, the results indicate that churn has only slight impact in BA and in ER networks.

5 Conclusion

We introduce Armonia, a unique flexible simulator framework that can be used to model dynamic interactions of nodes in complex networks. It offers a topology component for specifying the network structure, a resource and replication component for modeling the overall distribution of resources on the nodes of the network, a search component for specifying the algorithm used to locate resources, and a statistics component for gathering useful measurements. It is the first general-purpose framework of its kind. Our aim is to use Armonia to study the behavior of various search strategies in complex networks with churn, where nodes may join or leave the network dynamically. We have focused on probabilistic flooding strategies, although other search policies can be implemented effi-

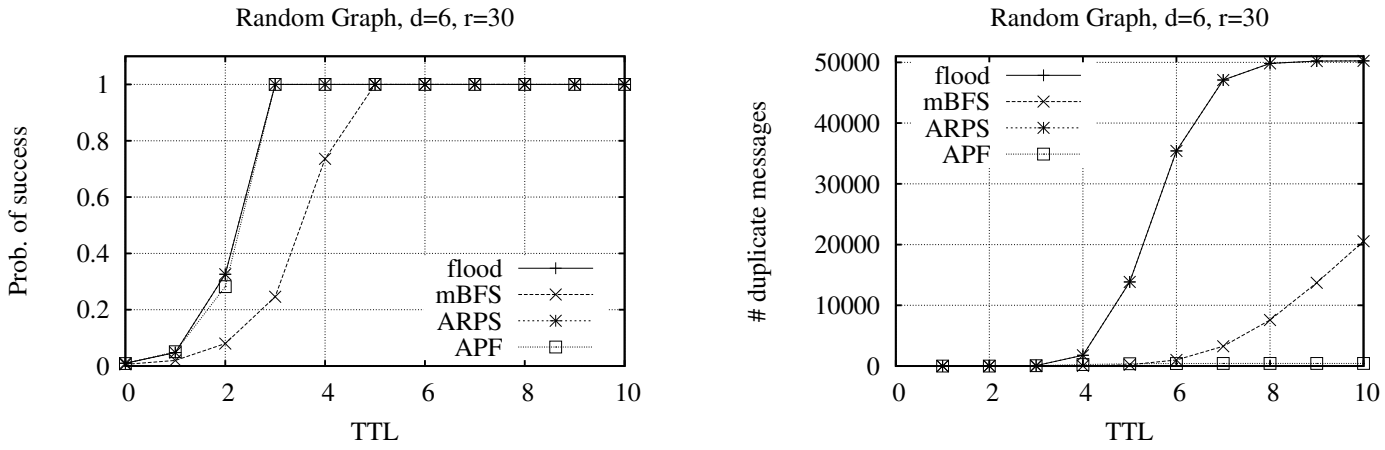

Figure 5. Probability of success (left) and duplicate messages (right) in an Erdos Renyi network of 25,000 nodes, with $d = 6$ and $r = 30$.

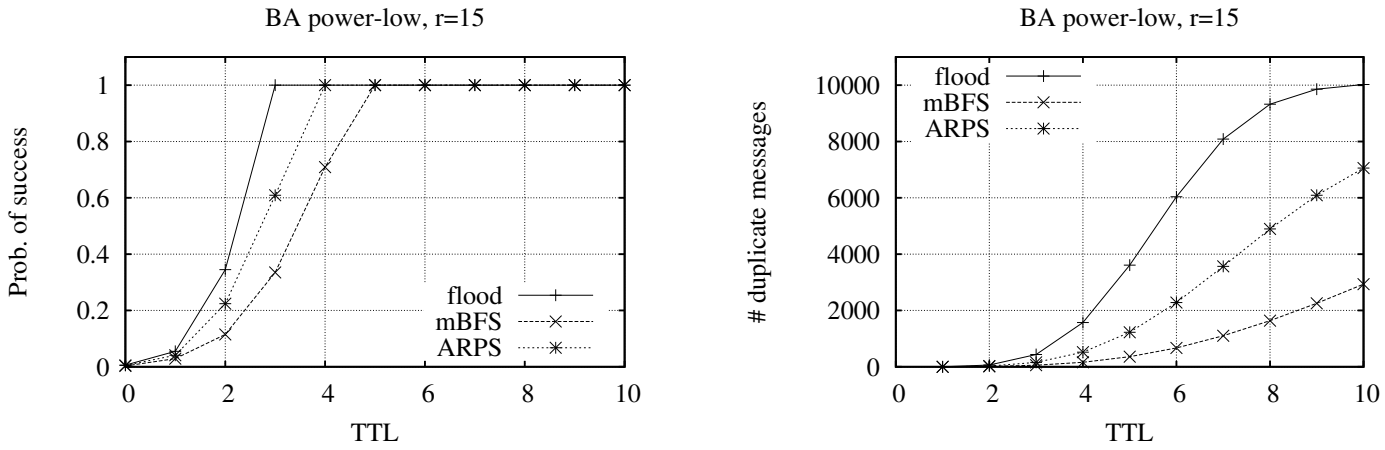

Figure 6. Probability of success (left) and duplicate messages (right) in a Barabási power-law network of 10,000 nodes, with $m = 2$ and $r = 15$.

ciently. Currently, we optimize our framework, while also adding new topologies and search strategies. Armonia will be available as an open-source project which we hope will serve the research community, as a useful simulation tool.

References

[1] M. Ángeles Serrano and M. Boguñá, "Tuning clustering in random networks with arbitrary degree distributions," *Physical Review Letters*, vol. 72, no. 3, p. 036133, Sep. 2005.

[2] M. Bastian, S. Heymann, and M. Jacomy, "Gephi: An open source software for exploring and manipulating networks," in *Proc. International AAAI Conference on Weblogs and Social Media*, 2009.

[3] A. Basu, S. Fleming, J. Stanier, S. Naicken, I. Wakeman, and V. K. Gurbani, "The state of peer-to-peer network simulators," *ACM Comput. Surv.*, vol. 45, no. 4, pp. 46:1–46:25, Aug. 2013.

[4] I. Baumgart, B. Heep, and S. Krause, "OverSim: A flexible overlay network simulation framework," in *Proc. 10th IEEE Global Internet Symposium (GI '07) in conjunction with IEEE INFOCOM 2007, Anchorage, AK, USA*, May 2007, pp. 79–84.

[5] S. Boccaletti, V. Latora, Y. Moreno, M. Chavez, and D.-U. Hwang, "Complex networks : Structure and dynamics," *Phys. Rep.*, vol. 424, no. 4-5, pp. 175–308, Feb. 2006.

[6] S. Carmi, R. Cohen, and D. Dolev, "Searching complex networks efficiently with minimal information," *EPL (Europhysics Letters)*, vol. 74, no. 6, p. 1102, 2006.

[7] Y. Chawathe, S. Ratnasamy, L. Breslau, N. Lanham, and S. Shenker, "Making gnutella-like p2p systems scalable," in *Proc. 2003 Conference on Applications, Technologies, Architectures, and Protocols for Computer Communications*, ser. SIGCOMM '03. New York, NY, USA: ACM, 2003, pp. 407–418.

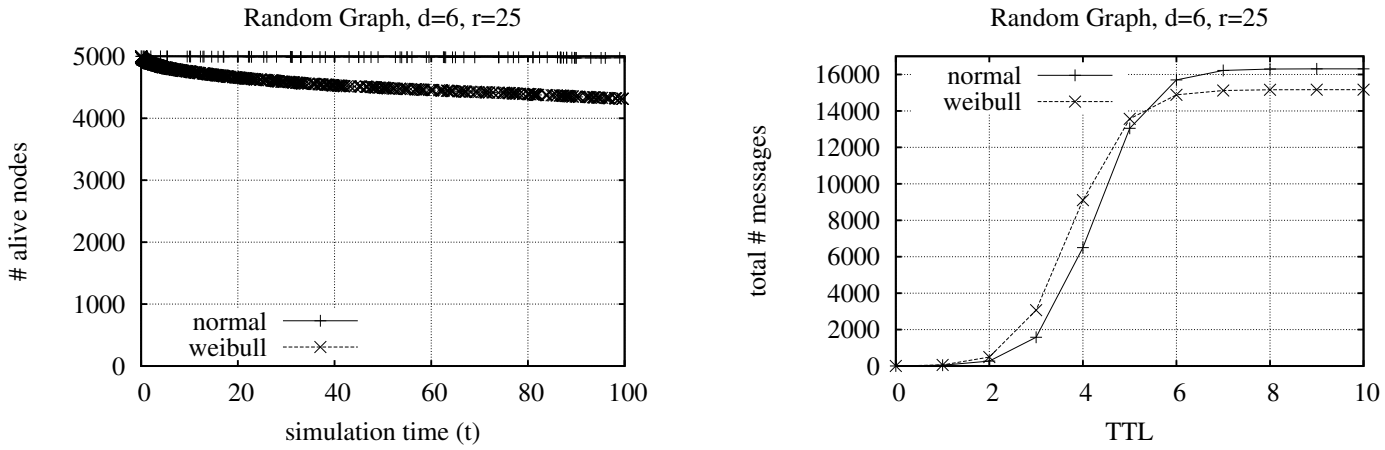

Figure 7. A sample of live nodes for the first 100sec of the simulation (left) and the total number of messages (right) under flooding in an ER network. We consider churn with 5,000 initial nodes.

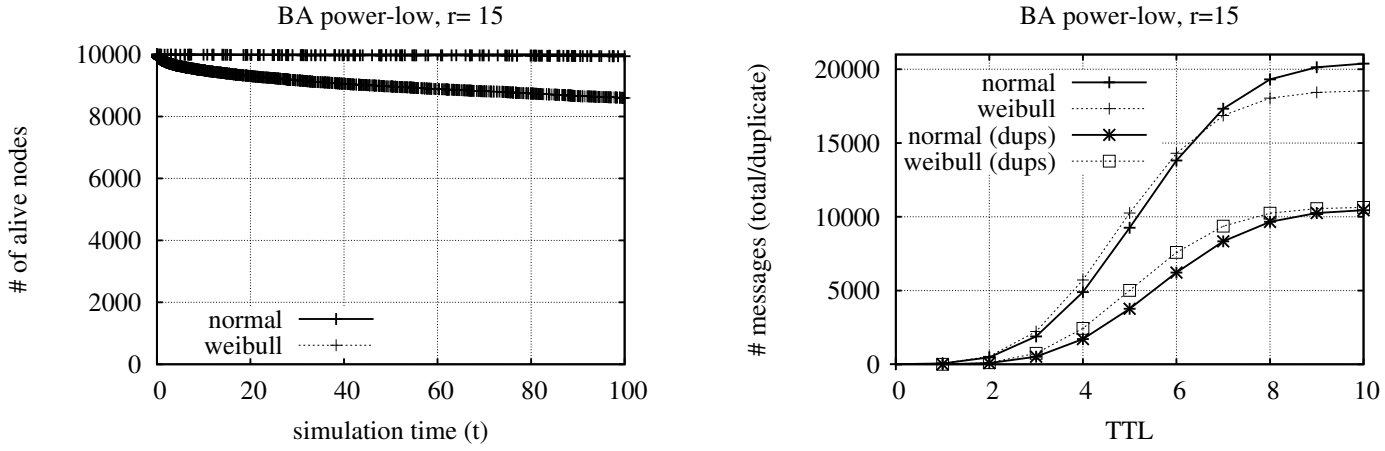

Figure 8. A sample of live nodes for the first 100sec of the simulation (left) and the number of messages (right) under flooding in a Barabási power-law network with 10,000 initial nodes; we show both the total number of messages and the number of duplicates.

[8] P. Erdös and A. Rényi, "On the evolution of random graphs," *Publ. Math. Inst. Hung. Acad. Sci*, vol. 5, pp. 17–61, 1960.

[9] C. Gkantsidis, M. Mihail, and A. Saberi, "Hybrid search schemes for unstructured peer-to-peer networks," in *Proc. INFOCOM 2005, 24th Annual Joint Conference of the IEEE Computer and Communications Societies*, vol. 3, march 2005, pp. 1526 – 1537 vol. 3.

[10] K. V. Jónsson, Y. Vigfússon, and O. Helgason, "Simulating large-scale dynamic random graphs in omnet++," in *Proc. 5th International ICST Conference on Simulation Tools and Techniques*, ser. SIMUTOOLS '12. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2012, pp. 311–318.

[11] V. Kalogeraki, D. Gunopulos, and D. Zeinalipour-Yazti, "A local search mechanism for peer-to-peer networks," in *Proc. 11th International Conference on Information and Knowledge Management*, ser. CIKM '02. New York, NY, USA: ACM, 2002, pp. 300–307.

[12] J. Leskovec and R. Sosič, "SNAP: A general purpose network analysis and graph mining library in C++," http://snap.stanford.edu/snap, Jun. 2014.

[13] G. Liu, Y. Wang, M. Orgun, and H. Liu, "Discovering trust networks for the selection of trustworthy service providers in complex contextual social networks," in *Web Services (ICWS), 2012 IEEE 19th International Conference on*, June 2012, pp. 384–391.

[14] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker, "Search and replication in unstructured peer-to-peer networks," in *Proc.*

16th International Conference on Supercomputing, ser. ICS '02. New York, NY, USA: ACM, 2002, pp. 84–95.

[15] S. V. Margariti and V. V. Dimakopoulos, "On probabilistic flooding search over unstructured peer-to-peer networks," *Peer-to-Peer Networking and Applications*, vol. 8, no. 3, pp. 447–458, 2015.

[16] S. Mccanne, S. Floyd, and K. Fall, "NS2 (Network Simulator 2)." [Online]. Available: http://www-nrg.ee.lbl.gov/ns

[17] A. Montresor and M. Jelasity, "PeerSim: A scalable P2P simulator," in *Proc. of the 9th Int. Conference on Peer-to-Peer (P2P'09)*, Seattle, WA, Sep. 2009, pp. 99–100.

[18] S. Naicken, A. Basu, B. Livingston, S. Rodhetbhai, and I. Wakeman, "Towards yet another peer-to-peer simulator," in *Proc. 4th International Working Conference on Performance Modelling and Evaluation of Heterogeneous Networks (HET-NETs)*, Ilkley, UK, August 2006.

[19] S. Naicken, B. Livingston, A. Basu, S. Rodhetbhai, I. Wakeman, and D. Chalmers, "The state of peer-to-peer simulators and simulations," *SIGCOMM Comput. Commun. Rev.*, vol. 37, no. 2, pp. 95–98, Mar. 2007.

[20] M. E. J. Newman, "Random Graphs with Clustering," *Physical Review Letters*, vol. 103, no. 5, p. 058701, Jul. 2009.

[21] S. Rhea, D. Geels, T. Roscoe, and J. Kubiatowicz, "Handling churn in a dht," in *Proc. Annual Conference on USENIX Annual Technical Conference*, ser. ATEC '04. Berkeley, CA, USA: USENIX Association, 2004, pp. 10–10.

[22] R. van der Hofstad, *Random Graphs and Complex Networks. Vol. I*. Lecture Notes, Aug. 2015. [Online]. Available: http://www.win.tue.nl/~rhofstad/NotesRGCN.html

[23] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proc. 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, ser. Simutools '08. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 60:1–60:10.

[24] D. J. Watts, "Networks, dynamics, and the small-world phenomenon," *American Journal of Sociology*, vol. 105, pp. 493–527, 1999.

[25] T. White, *Hadoop: The Definitive Guide*. O'Reilly Media, Inc., 2012.

[26] H. Zhang, L. Zhang, X. Shan, and V. Li, "Probabilistic search in p2p networks with high node degree variation," in *ICC '07, IEEE International Conference on Communications, 2007*, june 2007, pp. 1710 –1715.