The Journal of Systems and Software 100 (2015) 211–232

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

D-P2P-Sim+: A novel distributed framework for P2P protocols performance testing☆

S. Sioutas[a,*], E. Sakkopoulos[b], A. Panaretos[a], D. Tsoumakos[a], P. Gerolymatos[a], G. Tzimas[d], Y. Manolopoulos[c]

[a] Ionian University, Greece
[b] University of Patras, Greece
[c] Aristotle University of Thessaloniki, Greece
[d] Technological Educational Institute of Western Greece, Greece

ARTICLE INFO

Article history:
Received 3 April 2014
Received in revised form 31 October 2014
Accepted 1 November 2014
Available online 8 November 2014

Keywords:
Distributed storage systems
IoT and Web 2.0 applications
P2P data management

ABSTRACT

In recent technologies like IoT (Internet of Things) and Web 2.0, a critical problem arises with respect to storing and processing the large amount of collected data. In this paper we develop and evaluate distributed infrastructures for storing and processing large amount of such data. We present a distributed framework that supports customized deployment of a variety of indexing engines over million-node overlays. The proposed framework provides the appropriate integrated set of tools that allows applications processing large amount of data, to evaluate and test the performance of various application protocols for very large scale deployments (multi million nodes–billions of keys). The key aim is to provide the appropriate environment that contributes in taking decisions regarding the choice of the protocol in storage P2P systems for a variety of big data applications. Using lightweight and efficient collection mechanisms, our system enables real-time registration of multiple measures, integrating support for real-life parameters such as node failure models and recovery strategies. Experiments have been performed at the PlanetLab network and at a typical research laboratory in order to verify scalability and show maximum re-usability of our setup. D-P2P-Sim+ framework is publicly available at http://code.google.com/p/d-p2p-sim/downloads/list.

1. Introduction

Our era can be characterized by an explosion in the production of data: Internet of Things (IoT) and Web 2.0 technologies contribute to the production of increasing volumes of information that need to be stored, indexed and queried.

The Internet of things is a vision of the future Internet that expands beyond the virtual world to the physical word, through devices and wireless networks that enable the ubiquitous collection and transmission of information from uniquely identifiable objects. In (Atzori et al., 2010) the authors explain that the Internet of Things (IoT) cannot be restricted to just the application of RFIDs technology instead, it is a wider vision where devices, network and service technologies are combined introducing a new era of ubiquity. Devices with autonomous and proactive behavior, context awareness, capable of collaborative communications will contribute to IoT. In this plethora of information sources, scalability and personalization are necessary to handle massive heterogeneous resources as well as to enhance user-experience and allow access to the most relevant parts of the data. With respect to personalization according to (Personalization Reports, n.d.) over 80% of users prefer the numerous personalization services that businesses and social sites offer.

Several problems have been identified in the literature that has to be addressed in order to realize the vision of IoT. One of them is scalability. As mentioned in Miorandi et al. (2012), the management of information and knowledge is creating scalability problems because for every entity in the physical world a virtual entity has to be built, moreover, massive services execution is required that need to handle massive heterogeneous resources. For example, RFIDs technology is being introduced in Supply Chain Management, to

☆ A very preliminary version of this work was presented in ACM SAC'12, pp. 853–858, March 25–29, 2012, Riva del Garda, Italy. This work was partially supported by Thales Project entitled "Cloud9: A multidisciplinary, holistic approach to internet-scale cloud computing". For more details see the following URL: https://sites.google.com/site/thaliscloud9/home.

* Corresponding author.
E-mail address: sioutas@ionio.gr (S. Sioutas).

http://dx.doi.org/10.1016/j.jss.2014.11.001
0164-1212/

increase visibility, data quality and efficiency and better satisfy customers. As a result huge number of data is being generated. These large amounts of distributed data have to be exploited in a constructive way, they need for example to be analyzed and matched to the organization strategy. Constant monitoring of goods, real-time sales information collection and processing can benefit Supply Chain Management, allowing taking action in time and decreasing costs. Building a system to store and process this information adequately is challenging technically and financially (Zhang et al., 2011).

Consider for example an infrastructure for storing and processing data collected from sensors like RFIDs that participate in the supply chain. Each member of the supply chain is interested in asking different questions in different sets of data. Personalization is utilized in IoT Supply Chain Management environment in different ways. A participant may play different roles based on the location and the device used hence different participant profiles are necessary. Managing digital identities and user profiles in IoT environments is problem that needs to be addressed in a large scale.

The sheer size of both data and their diverse sources render centralized, legacy systems inept at storing and processing them. The desired scalability and efficacy to handle the advanced processing required are provided by distributed solutions as both academic and business innovation has already indicated. Recently P2P storage solutions have been adopted to address these issues.

Designing new protocols and overlays for P2P systems includes apart from the mathematic proof of their correctness and efficiency, and experimental measurements that confirm the theoretical results. In the case of P2P, the experimental study in real environment is most times practically impossible, because of the enormous volume of data and involved resources that are required. For this reason simulation environments are employed in order to approach best real life conditions.

In this work, we present a framework that is able to efficiently and scalably support millions of physical cooperating nodes running different protocols and diverse personalized datasets on different levels of granularity. We present the extension of D-P2P-Sim, the so-called D-P2P-Sim$^+$ framework that is able to deliver and to provide two key features:

First, D-P2P-Sim$^+$ supports and facilitates the deployment of very large-scale (multiple million) P2P node systems. It also includes simple, easy and organized tools to achieve the collection of numerous statistics and execution of web-scale concurrent queries, i.e., implementation of node failure and node departure recovery strategies for best simulation of real life conditions.

Second, its modular design enables the use of multiple indexing/processing engines, allowing different applications or even groups of users within the same application to customize their processing needs. All these new features are integrated in the same user friendly environment that provides extensive statistical results. The D-P2P-Sim$^+$ system can simulate 600,000+ nodes in a stand-alone environment and millions of nodes as it can be executed in a distributed mode within a researcher's laboratory. In this sense, it takes advantage of the available resources from multiple computers that usually exist in a typical laboratory or a research network.

D-P2P-Sim$^+$ brings forward appropriate tools and ready to use infrastructure in order to facilitate the researchers' tasks in-line with the framework. Multiple protocols have been simulated in order to show that it takes couple of hours to initialize a network of 100,000 nodes at a middle range configured PC. Moreover makes available a framework so that researchers can develop, share and re-utilize failure detection and handling scenarios during their research on P2P network protocols.

Distributed simulations with D-P2P-Sim$^+$ can be designed and delivered with the same framework in different environments but with the same single setup. We show that only a small scale laboratory environment (5 computing machines) is an adequate environment to deliver multi-million node simulations at ease. Furthermore, we present the roadmap and demonstrate the results to deliver a small scale laboratory setup to a successfully verified experiment of six (6) protocol evaluations at the PlanetLab platform (http://www.planet-lab.org) with limited effort given resources available. PlanetLab constitutes a world wide research network that supports the growth and implementation of new services for networks.

The rest of the paper is as follows. Section 2 discusses related work and motivation. Section 3 presents the architectural details in depth. Next, Section 4 discusses failure scenarios and recovery strategies on large number of nodes networks. Section 5 discusses experimental results showing examples for P2P networks with up to 2 Million nodes using a variety of lookup protocols. Section 6 shows that D-P2P-Sim+ is easily executed in the PlanetLab environment without need to change the protocol. Finally, Section 7 concludes the paper.

2. Motivation

The following subsections describe the reasons motivating us to design a novel distributed framework.

2.1. Handling IoT large amounts of data

RFIDs technology is being introduced in Supply Chain Management, for constant monitoring of goods, real-time sales information collection and processing. As a result huge number of data is being generated. Building a system to store and process this information adequately is a challenging technically and financially.

As an example suppose that an international company that transports goods tagged with RFIDs is storing all information received by all tags product position, movement, description, etc. Every product and its movement is an entry to the system. The company is transporting millions of goods through several locations and the information and its history needs to be stored. In such cases the administration might be interested to know for a specific product the number and places of delay for all product transportation within the last years. In such case each product is a tuple (ProductID, location, time, expected_time (derived attribute), description). Each tuple might reside in different node, based for example on the location of the RFID reader that collected it. Based on the information, location of product delays can be identified and further examined:

SELECT ProductID, location
FROM Nodes
WHERE time > expected_time

As another example we assume that the company stores information regarding all customers that are using their mobile phone to buy products worldwide. Each customer is a node that maintains information as a tuple (UID, number_of_items_purchased, time_of_purchase, location_of_purchase, profile_user). Tuples are stored based on the place of purchase. As customers are purchasing goods information is inserted to the system. The company would be interested to find answers to queries like:

SELECT COUNT(UID)
FROM Nodes
WHERE number_of_items_purchased >3
AND time_of_purchase in $[t_1,t_2]$ *AND product = productID*

Not only querying the information is crucial for the participants of a supply chain but also taking the results on time. Major decisions, i.e., about purchasing, transporting and producing goods are based on the information retrieved. Moreover, usually participants in different stages of the supply chain as well as customers have signed service level agreements in order to guarantee the timely delivery of goods and their quality. The violation of these quality agreements result in a penalty to compensate. The timely extraction of useful information from the data is hence of paramount importance.

That which would seem a trivial query for a traditional database, is not so, when enormous sets of events/data have been produced from distributed devises. Usually in these environment a new approach in storing and processing the data is adopted: P2P systems as a storage solution provides low cost in sharing of the supply chain information while each node participating in this sharing can be independent, autonomous and use privacy preserving mechanisms to define visibility of data. Range queries performance in a P2P network can vary therefore several overlay protocols have been proposed that improve performance based on different factors imposed by applications and data. The selection and/or creation of the appropriate node based protocol is crucial because it directly influences the response time of the queries hence the satisfaction of the SLAs. Experiments need to be done to decide the appropriate P2P solution settings, and such experiments are not feasible in real environments because of the enormous data collected and the required resources.

Hence, in such IoT applications, where we collect information from several resources that are distributed and each one has an independent set of information, we need tools that can facilitate the development of distributed systems capable of handling queries addressing multi-million nodes. The most prominent way to make decisions upon building such systems is the use of simulators.

In this paper we propose a simulation environment that contributes in taking decisions regarding the choice of the protocol in storage P2P systems. Our contribution is that besides scalability our simulator provides a systematic mechanism to gather statistics. Moreover it simulates a real life P2P storage system realistically taking into consideration nodes departure and failures thus producing accurate results.

2.2. IoT and personalization

IoT and Web 2.0 technologies (Brusilovsky et al., 2007; Brusilovsky and Maybury, 2002) make it possible to deliver personalized views or versions of web data, improving usability and thus productivity.

Adaptation features enable personalization of the interface and the flow of business processes to the different characteristics and role of each internal user. The user profile records information concerning the user and his knowledge state based on the location, the device, the preferences and other information collected by different sensors. User profiles can determine personalized views and they can be distributedly stored across several servers.

This information is vital for the system's operation according to the user's needs and preferences. However, management, recording and manipulation of user profiles at large scale Web Systems can be tedious as multiple replication of Web Systems does not coherently support synchronization of user activity (Wang et al., 2008). Users transparently enter the Web System at different servers and, as a consequence, their profile can be distributedly stored across several servers.

The explosion in adoption of Web data modeling (e.g., XML and XML Schema (Bertino et al., 2004)) has brought forward critical issues in the process of knowledge personalization in Web Services. In this context, service personalization is another issue to be faced, with fine-grained user management being strongly required.

The process of combination for fine-grained multidimensional user models with knowledge representation and management techniques for making Web Services knowledge-aware usually involves a number of Web Services distributed in a network of numerous nodes (Cuzzocrea, 2006). Thus, a distributed approach would be highly useful to efficiently process the amount of profile data.

3. Related work

Contemporary distributed systems involve very large populations of commodity nodes independent of the specific underlying architecture: BitTorrent has reported stats where over 20 million daily active users download over 400,000 torrents on average and the number of users is already 100+ millions. Other P2P-based networks like Live Messenger report 300+ millions of monthly active users online (Windows Live Messenger, 2010) and 25+ million nodes (peak online time) in the Skype chat network (A. S. Report on Skype and Usage, 2011). On the other hand, the processing and retrieval needs per application range vastly: Social and web advertising sites perform map-reduce based intensive processing to identify trends and mine useful information (e.g., Shmueli-Scheuer et al., 2010), while real-time applications or simple analytics tools require efficient point-range-aggregate queries over bulk personalized datasets (e.g., Chen et al., 2011). Thus, it is also imperative that, besides the required scalability, the platform should be modular enough to provide support for different types of storage/processing engines.

P2P simulators survey (Naicken et al., 2007) shows that the majority of the cases studied involved a number of different custom simulation tools. The survey finds that custom made simulators (developed per protocol or in some case utilized by a particular research group) far outnumbers the use of known ones. Resulting, comparison and replication of evaluation procedures and outcomes is very difficult. Additionally, developing multiple times the same protocols from the beginning is needed. A key point discussed in the survey is that 'the poor state of existing P2P simulators is the reason that much published research makes use of custom built simulators'. The main drawbacks of the simulation mechanisms found were (i) no systematic mechanism to gather statistics of the simulation executions, and (ii) absence of graphical user interface support to support user friendliness. To cope with the above problem, the P2P simulator (Sioutas et al., 2009) has been presented.

D-P2P-Sim includes extensive and effective mechanism for statistics as well as an effective and easy to use GUI. In this work we extend D-P2P-Sim.

The analysis of Naicken et al. (2007) shows that P2P simulators do not usually support the creation of more than 10,000 nodes (Jagadish et al., 2006). Only a limited number of cases conducts experiments with more nodes (Cowie et al., 1999), however even then no more than a few 100,000 P2P nodes, mainly because of memory requirements. Even though there is evidence that since 2007 a greater proportion of research is being tested with real systems and real trace data (Basu et al., 2013), the scalability of P2P simulators has not been shown in wide scale. Recently, an extension of Peersim (Montresor and Jelasity, 2009) has been presented (Mayer et al., 2012), showing that it is possible to schedule parallel executions of Peersim in order to take advantage of multiple cores within a processor. However, the limitations of the maximum number of nodes remains in each Peersim execution instance. Moreover, no results on the scalability of the proposal (Mayer et al., 2012) have been shown yet to the authors' best knowledge. SimGrid (Casanova et al., 2008; Quinson et al., 2012; Bobelin et al., 2012) has

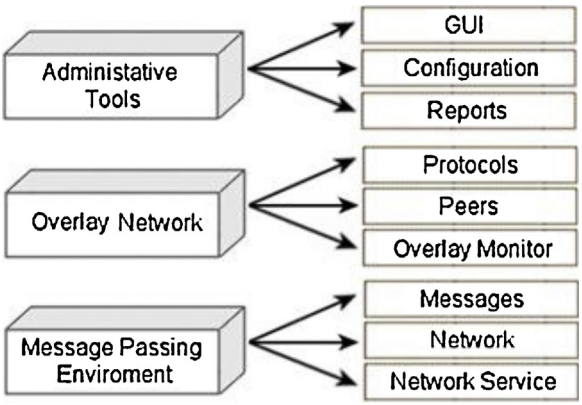

Fig. 1. Simulator main packages.

been reported with results of 2 million nodes on a 12-core processor; this however brings the simulated number of nodes below of 200,000 P2P nodes per processor core. In the SSNG simulator (Teng et al., 2014) the maximal number of (super) peers achieved has been up to 1 Million using the available hardware. The presented framework, using lightweight and efficient collection mechanisms can easily reach >600,000 P2P nodes in a typical Pentium 4 standalone PC and multi-million P2P node in a small network that every research laboratory has.

P2P simulators fail to include real-time registration of multiple measures as well as inline features to simulate easily and seamlessly (a) randomized node departures and (b) node failures (and node failure scenarios) to depict more realistically real life conditions (Naicken et al., 2007; Cowie et al., 1999). A work on complex queries has been presented using a hybrid overlay that integrates a structured P2P system with an unstructured one (Lai and Yu, 2012). Researchers usually customize simulators and deliver separate development to facilitate such experimental evaluation (Jagadish et al., 2006).

4. Basic architecture

In this section, the basic architectural features are presented in depth. The figures following depict the main parts of the framework environment, how the packages are related and the design decisions that have been made. The modular architecture allows the researcher to develop quickly and using a clear approach every new protocol that one might want to test on large scale simulations. The simulator is organized in packages as shown in Fig. 1. All these new features are integrated in a user friendly environment that provides extensive statistical results following the paradigm of (Sioutas et al., 2009). Details on failure recovery strategies, statistics and GUI will be given in the following sections.

The basic architecture of the Java P2P simulator is depicted in more details in Fig. 2. Four (4) different modules are depicted:

1. The Message Passing Environment (pink color)
2. The Overlay Network and protocol instance that is implemented each time (yellow color)
3. The Remote Services in order to enable distributed execution and simulation (green color)
4. Statistical Tools and Administrative Management (orange color)

Peers are autonomous and independent network-entities that execute protocol tasks. Peers communicate each others via messages in the overlay network. Since each peer is implemented as a runnable task, it can execute tasks in asynchronous and parallel mode. Each overlay-peer is not always in running or "execute" mode. On the contrary, a parameterized thread pool is used in order to ensure the high performance of simulator as well as to avoid possible bottlenecks. The high performance is ensured since (a) we do not have to pay an extra cost in order to create a new thread and (b) we do not have to consume extra resources when a thousand of threads is in "busy wait" mode. Threads are in this mode when there is no task to execute.

When the D-P2P-Sim+ is running in distributed environments, the responsibility for creating and managing the overlay network is also distributed in different systems. Each such a system is named "application node" and runs an instance of simulator. Application nodes are running in parallel and independently via a "remote method invocation" java mechanism.

The basic architecture of the Java P2P simulator presented in Fig. 2 is based on the message passing environment. The environment is solidly based on peers exchanging messages in order to build the overlay network and to carry out the search, insert and delete operations in the P2P network.

Our implementation strategy has determined Message and Data separation of concerns and as a result respective classes have been designed and developed. Any Message object holds the network device/sensor id either serving as a transmitter or receiver. Additionally, the operation that the message serves is also kept within the object. In our solution any Message is coupled to a Data object that includes all its valuable information.

Furthermore, the network's behavior is simulated through dedicated Network class objects. The idea of this object is to deliver the streaming behavior of networking formulating, maintaining and orchestrating buffers, i.e., a type of ad-hoc queues of Messages. Any Network object is able to send and receive Messages of different flavors, i.e., broadcasting, messages to a specific Node or group of Nodes to serve all possible needs during simulation setups. As it is expected, all different types of messages (i.e., "search", "insert" or "delete" operation messages) "travel" within a Network Object. The simulator proposed, for any given message in the network, is able to keep a log record keeping track of its type, sender and recipient to support further statistical analysis if needed. The produced log messages are used by the graphical user interface to show how an operation is incrementally completed.

The network is responsible to notify a node of an incoming message and also is able to reply to queries about messages in awaiting lists (the buffers already mentioned above). The broadcasting message may be used during construction steps of the overlay under simulation in order to send initialization messages to ranges of peers if needed. Broadcast is not used during any search operation in order to avoid flooding consequences.

4.1. Message passing environment

The message passing environment (see the pink boxes of Fig. 2) emulates the physical underlying network and consists of the following modules: Message, Network, Network Monitor and Network Filter. The class "Network" implements a priority queue as well as two methods for inserting and deleting messages to or from the queue. These methods are named "sendMsg" and "recvMsg", respectively. They are synchronized and use the appropriate "mutex key" for accessing under race conditions the queue, which is a shared resource (or variable), in a "mutual exclusive" mode. An overlay-peer sends and receives messages by calling "sendMsg" and "recvMsg" methods, respectively. The exchanged massages have the identities (IP addresses) of sender and receiver, respectively, as well as additional information concerning the current number of consumed hops. This Info-Field of class "Message" is user defined by implementing the appropriate "BodyMessage" interface. This interface defines one an only method ("getType"), which returns the type of message. The "NetworkMonitor" thread is running repeatedly

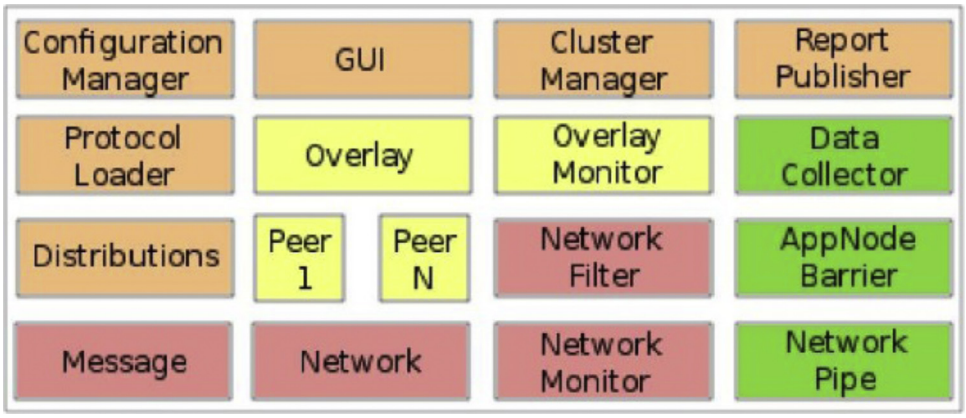

Fig. 2. The D-p2p-sim+ architecture.

and checks the network queue for the existence of new threads that waiting for serving. If there is such a thread, the "Network-Monitor" finds the destination-peer to which this message has to be transferred. If the queue is empty, the "NetworkMonitor" thread becomes idle till at least one message arrives in the network queue. In this way no extra network resources have to be consumed. Moreover, the "Network Filter" class refines a message copy in order to extract useful statistics.

4.2. The overlay network

The "overlay network" consists of the following modules: Peer 1, Peer2,. . .,PeerN, "Overlay Monitor" and "Overlay" (see the yellow boxes of Fig. 2). The basic role of "overlay network" is to initialize a p2p structure with a user-defined number of peers by repeatedly calling the appropriate "node-join" method. Then each peer is in sleep (idle) mode and weak up when there is a message for it. Moreover each peer can run lookup, insert and delete operations according to a specific p2p protocol. If a message for a specific peer is detected by "Network Monitor", then this peer receives the appropriate notification from "Overlay Monitor" class.

4.3. Remote services

"Remote-services" (see the green boxes of Fig. 2) architecture is required for supporting distributed execution and consists of the following modules: "Network Pipe", "Data Collector" and "AppNode Barrier". When a message-receiver concerns another application node, this message must be deleted from the local queue and inserted to the queue of the application-node, which is responsible for the node-receiver. The "NetworkMonitor" thread based on the "NetworkPipe" method (class), inserts the message to the queue of remote node. "NetworkPipe" interface satisfies the requirements defined by the RMI (Remote Method Invocation). In this way, we extend the overlay-peers communication to an application-nodes communication. The synchronization between threads is implemented in "AppNode Barrier" class that calls three methods: "enable", "disable" and "isEnabled". The "enable" class enables the barrier and results in thread freezing. The "disable" class disables the barrier and results in thread re-freezing. As a result, the thread is re-executed from the point in which had stopped running. The "isEnabled" class returns the thread's current status. The communication of application nodes is also required for

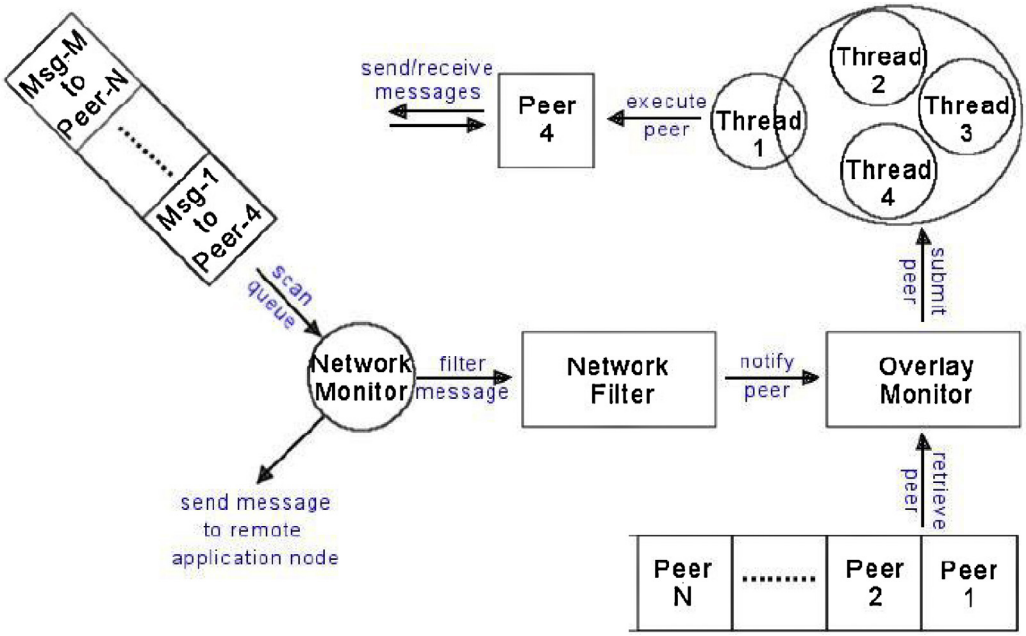

Fig. 3. The "application node" architecture.

```
< distribution>
  < random>
    < seed>1</seed >
< /random>
                < beta>
   < alpha>2.0</alpha >
    < beta>4.0</beta >
< /beta>
< powerLaw>
   < alpha>0.5</alpha >
   < beta>1.0</beta >
< /powerLaw>
< /distribution>
```

Fig. 4. Snippet from config.xml with the pre-defined distribution's parameters setup.

the transfer of partial simulated-results collected by each of them. "Data Collector" module has been implemented for this purpose. In order the remote-access services to be activated, each application node has to bind an instance of each remote service to an appropriate RMI record.

The "Application Node" architecture is depicted in Fig. 3.

4.4. The "administrative-tools"

To support a GUI for multiple protocol implementations and to allow wide customization in testing scenarios and collection of metrics, a number of *administration tools* have been packaged into the simulation framework. *GUI* allows the high level researcher and protocol designer to perform protocol testing without involving source code at all. Furthermore GUI facilitates validation of protocols by independent researchers in an easy and straight forward UI functionality that incorporates collection and presentation of metrics. Admin tools have specifically been designed to support reports on a collection of wide variety of metrics including, protocol operation metrics, network balancing metrics, and even server metrics. Such metrics include frequency, maximum, minimum and average of: number of hops for all basic operations (lookup-insertion-deletion path length), number of messages per node peer (hotpoint-bottleneck detection), routing table length (routing size per node-peer) and additionally detection of network isolation (graph separation). All metrics can be tested using a number of different distributions (e.g., normal, weibull, beta, uniform, etc.).

Additionally, at a system level memory can also be managed in order to execute at low or larger volumes and furthermore execution time can also be logged. The framework is open for the protocol designer to introduce additional metrics if needed. Furthermore, XML rule based *configuration* is supported in order to form a large number of different protocol testing scenarios (see Fig. 4). It is possible to configure and schedule at once a single or multiple experimental scenarios with different number of protocol networks (number of nodes) at a single PC or multiple PCs and servers distributedly.

In more details, the "administrative-tools" (see the orange boxes of Fig. 2). Architecture consists of the following modules: "Protocol Loader", "Configuration Manager", "GUI", "Report publisher" and "Distributions". It reads the configuration file, loads the user-defined p2p protocol by executing the "Protocol Loader" class and initializes all the other parts in order the emulation-procedure to be started. In particular, it executes the following steps:

1. It reads the running parameters
2. It Initializes the following emulators:
   a. Network
   b. NetworkMonitor
   c. PeerToPeerOverlay
   d. OverlayMonitor
3. It loads the protocol instance that has to be emulated
4. It creates the overlay network
5. It executes the experiments
6. It creates reports with useful statistics

The "Configuration Manager" class is reading and checking the validity of parameters. The distributed management of cluster is executed by the "Cluster Manager" class, which reads information about the cluster nodes via the appropriate configuration file. Each "application node" is described from an identity, the IP address as well as the overlay-part (neighbor nodes) that manages.

5. Overlay scalability

Apart from the fact that a number of packages facilitate development, a number of different protocols are available as sample code and executables in order to allow familiarization with the development and usage of the simulator: Chord, Baton*, Nested Balanced Distributed Tree (NBDT) (Sioutas, 2008), NBDT* (Sioutas, 2008), R-NBDT* (Sioutas, 2008) with advanced load distribution and ART (Sioutas et al., 2013). Since the load-balancing factor is very crucial for our applications (IoT and Web 2.0), we don't study randomized p2p protocols, like skip-lists and its variants (Pugh, 1990; Aspnes and Shah, 2007). This is exactly the rationale behind the selection of these six (6) protocols corresponding to DHT's (Chord like structures) and HTSs (Hierarchical Tree Structures like BATON and its variants) structures, which achieve excellent load balancing. For comparison purposes, an elementary operation's evaluation is presented in Table 1 between the above protocols. Moreover, the respective abstract classes and programming steps are depicted also at a simplistic *dummy protocol* in order to guide the programmers that first use the simulation framework proposed. Fig. 5 depicts main performance measurements.

The framework is particularly designed to allow large scale experimental evaluation of node based protocols. The incorporation of metrics collection methods into the framework reveals the space and time consumption of each implemented p2p protocol and this fact minimizes both the memory needed (and frees up space for more nodes to be executed) as well as the response time of the p2p Handling method we finally chose. Moreover, distributed scenarios allow multiple computers to participate in the same experiment increasing radically the number of nodes simulated.

6. Node failure and departure strategies and statistics

In this section we present the theoretical problem formulation as well as the implementation aspects of the operations running node failures and departures, respectively.

6.1. Theoretical problem formulation

P2P networks could be modeled as directed Graphs $G(V,E)$, with V be the set of peers (vertices) and E be the set of links (edges). Each vertex is linked to neighbor vertices, via routing tables. In DHT's (Chord like structures) each vertex has $O(\log N)$ neighbors, where N is the number of nodes (or vertices). In HTSs (Hierarchical Tree Structures), the number of neighbors varies from $O(\log N)$ (BATON and its variants) to $O(N^{1/2^k}/\log^c N)$ (ART and its variants). Since the load-balancing factor is very crucial for our applications (IoT and Web 2.0), we do not study randomized distributed structures, like skip-lists and its variants (Pugh, 1990; Aspnes and Shah, 2007).

Table 1
Performance comparison between Chord (and its variants), BATON (and its variants), ART and Skip randomized structures: N is the number of peers, c is a positive constant $0 < c < 1$ and b is an exponential power of 2 (f.e.: 2, 4, 16,. . .).

P2P network architectures	Lookup/insert/delete key	Load balancing	Join/depart node
CHORD	$O(\log N)$	$O(\log N)$ expected amortized	$O(\log^2 N)$ expected with high probability (w.h.p.)
H-F-Chord (α)	$O(\log N/\log\log N)$	$O(\log N)$ expected amortized	$O(\log N)$
LPRS-Chord	Slightly better than $O(\log N)$	$O(\log N)$ expected amortized	$O(\log N)$
NBDT and its variants	$O(\log\log N)$	$O(N^{1/2})$	$O(1)$ expected w.h.p.
Strong Rainbow Skip Graphs	$O(\log N)$	–	$O(\log N)$ amortized
BATON	$O(\log N)$	$O(\log N)$ expected amortized	$O(\log N)$
BATON*	$O(\log_m N)$	$O(\log N)$ expected experimentally (there is no theoretical proof)	$O(m \log_m N)$
ART	$O(\log_{b^2} \log N)$ expected w.h.p.	$O(N^{1/4}/\log^c N)$	$O(\log\log N)$ expected w.h.p.

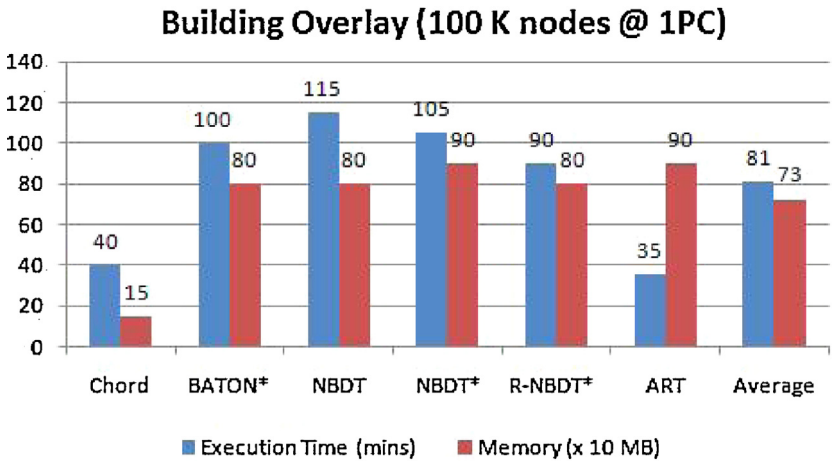

Fig. 5. Protocol support: execution time and memory requirements.

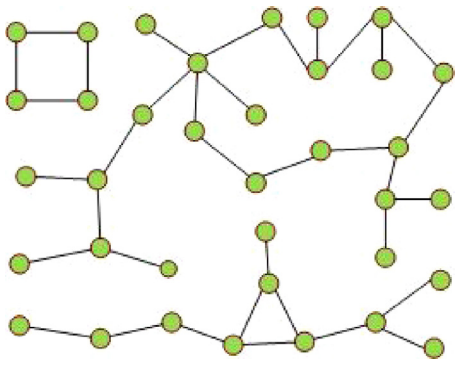

Fig. 6. A graph with three connected components.

Definition 1. In graph theory, a graph G is said to be k-vertex-connected (or k-connected) if it has more than k vertices and remains connected whenever fewer than k vertices are removed. The vertex-connectivity, or just connectivity, of a graph is the largest k for which the graph is k-vertex-connected.

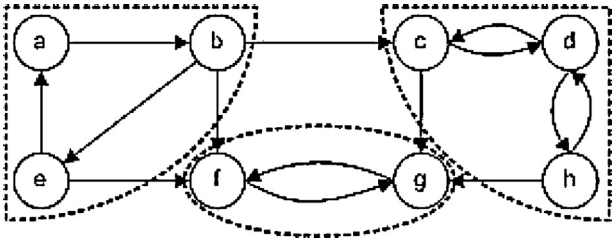

Fig. 7. Graph with strongly connected components marked.

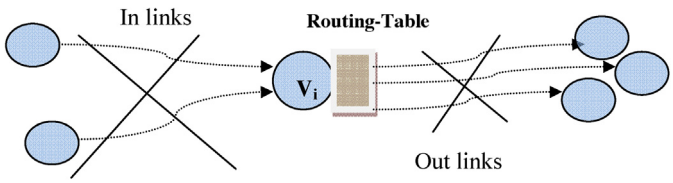

Fig. 8. $O(\log N)$ failures/departures of "out" links and $O(\log N)$ failures/departures of "in" links.

Definition 2. In graph theory, a graph is k-edge-connected if it remains connected whenever fewer than k edges are removed. The edge-connectivity of a graph is the largest k for which the graph is k-edge-connected.

Formal Definition 2. Let $G(V,E)$ be an arbitrary graph. If subgraph $G' = (V, E\backslash X)$ is connected for all $X \subseteq E$ where $|X| < k$, then G is k-edge-connected.

Definition 3. Minimum vertex degree gives a trivial upper bound on edge-connectivity. That is, if a graph $G(V,E)$ is k-edge-connected then it is necessary that $k \leq \delta(G)$, where $\delta(G)$ is the minimum degree of any vertex $v \in V$. Obviously, deleting all edges incident to a vertex v, would then disconnect v from the graph.

Definition 4. In graph theory, a connected component (or just component) of an undirected graph is a subgraph in which any two vertices are connected to each other by paths, and which is connected to no additional vertices in the supergraph. For example, the graph shown in the illustration below (Fig. 6) has three connected components. A graph that is itself connected has exactly one connected component, consisting of the whole graph.

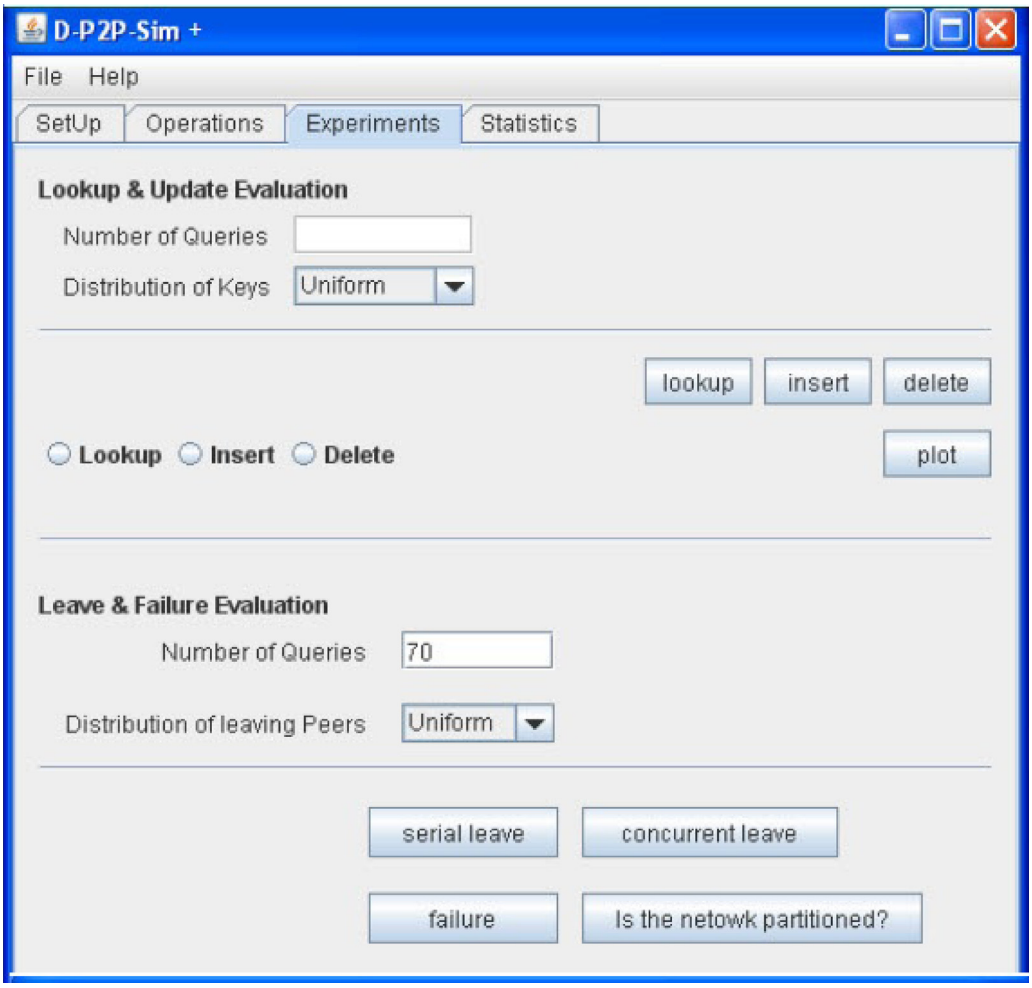

Fig. 9. Setup, operations, experiments and statistics option.

Definition 5. In the mathematical theory of directed graphs, a graph is said to be strongly connected if every vertex is reachable from every other vertex. The strongly connected components of an arbitrary directed graph form a partition into subgraphs that are themselves strongly connected (see Fig. 7). It is possible to test the strong connectivity of a graph, or to find its strongly connected components, in linear time.

Theorem 1. If the distribution of node failures or departures is not uniformly (f.e.: zipfian or powlow) distributed over the Graph vertices, then DHT's are $O(\log N+1)$-vertex and $O(\log N+1)$-edge connected.

Proof. Imagine the worst-case scenario where the whole set of failures or departures occurs in a single vertex Vi only, destroying all the "in" and "out" links (see Fig. 8). Then, after $2 * O(\log N)$ failures or departures, we will have two disconnected components: the single vertex Vi and the remaining graph. The theorem follows.□

Theorem 2. If the distribution of node failures or departures is not uniformly (f.e.: zipfian or powlow) distributed over the Graph vertices then the vertex and edge connectivity for HTSs varies from $O(\log N+1)$ (for BATON and its variants) to $O(N^{1/2^k}/\log^c N)$ (for Art and its variants), respectively.

Proof. We work in a similar way as in Theorem 1.□

But, what happens, if the departures/failures are uniformly distributed over the graph vertices? Let $G=(V,E)$ be the initial P2P Graph. Let $\#N_F$ be the number of node failures and $\#N_D$ be the number of node departures. Then:

$V_{NF} = \{V_i, 1 \le i \le \#N_F\}$ and $V_{ND} = \{V_J, 1 \le j \le \#N_D\}$.

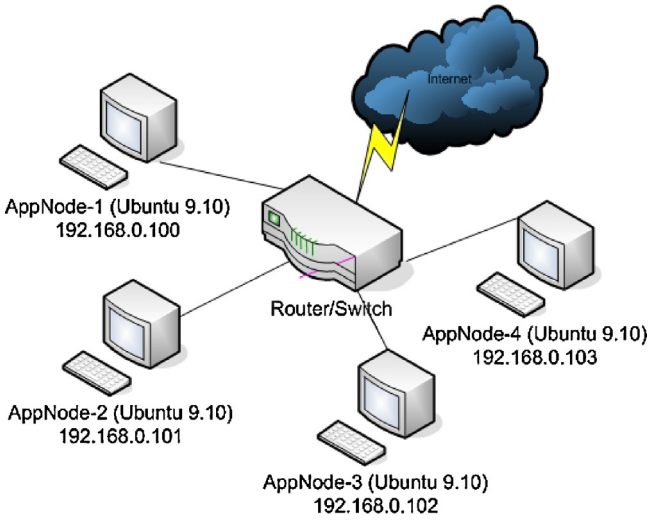

Fig. 10. The distributed environment.

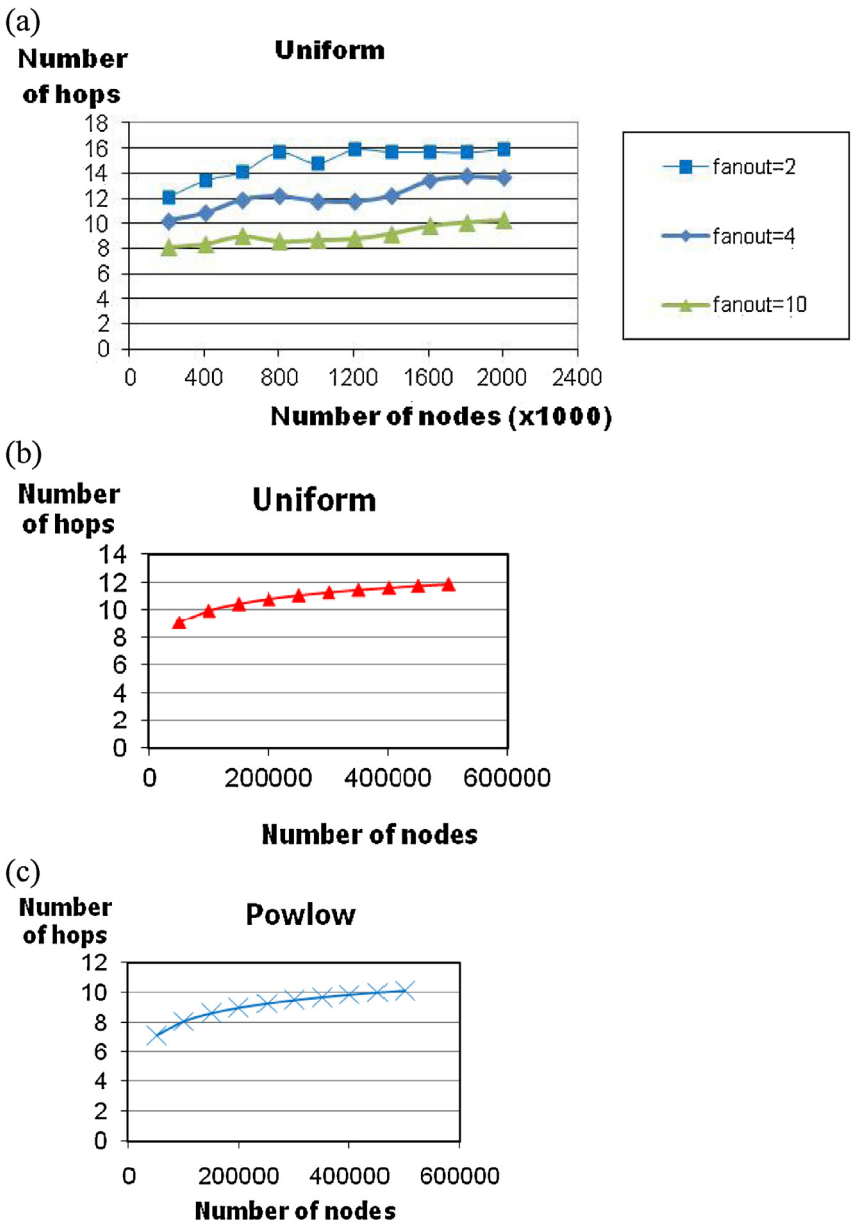

Fig. 11. (a) Baton* simulations with up to 2 Million P2P nodes: Lookup Cost Measure. (b) ART simulations with up to 600 K P2P nodes and Uniform Distribution: Lookup Cost Measure. (c) ART simulations with up to 600 K P2P nodes and powlow Distribution: Lookup Cost Measure.

Let also $\#e_f$ be the number of failed links (edges) caused by node failures and $\#e_D$ be the number of invalid links (edges) caused by node departures. Then:

$E_{NF} = \{e_K, 1 \le k \le \#e_F\}$ and $E_{ND} = \{e_m, 1 \le m \le \#e_D\}$.

As a result, $G_{NF} = (V_{NF}, E_{NF})$ and $G_{ND} = (V_{ND}, E_{ND})$ are the subgraphs that represent the failed/invalid parts of the whole graph G. Obviously, the resulted graph G' is defined as follows:

$G' = G - G_{NF} - G_{ND}$

The crucial question is: Is G' still connected?

As we have proved experimentally in Section 8, if $|V_{NF}| + |V_{ND}| < 25\%$ then DHTs are still connected. Also, HTSs are more resistant, since the whole graph remains connected if $|V_{NF}| + |V_{ND}| < (51.8 + 56.4\%)/2$. Of course, in both cases, the performance of searching – lookup operation deteriorates; however the graph connectivity is still remaining. More details will be described in Section 8.

6.2. *Implementation aspects of "node-failure" and "node-departure" operations*

Node failures and departures are important in order to achieve real life conditions during simulation. The presented framework supports new operations and services so that it provides services for node failure and network recovery and for node departures and substitutions. Simulators up to now tend to limit themselves to support (a) import of nodes for the creation of overlay network and (b) searching, (c) import and deletion of keys in nodes.

For this reason, we present the operations of a new simulator with all the necessary services to implement strategies for

- the self-willing departure of peer nodes (node departure)
- the failure or their sudden retirement for any reason (network failure, application failure, etc.)
- monitoring whether the overlay network is parted after successive node failures or departures

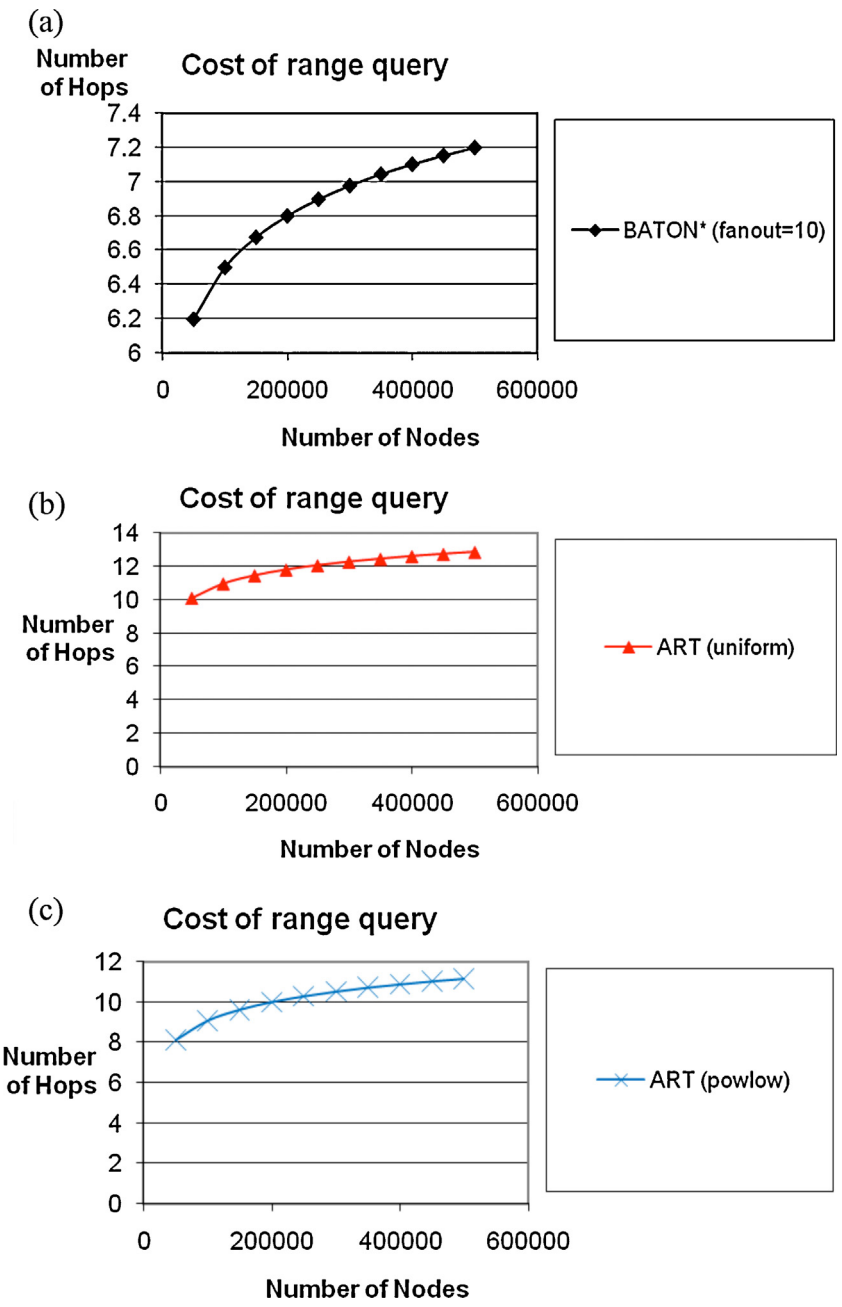

Fig. 12. (a) BATON* Simulations with up to 600 K P2P nodes: Range Query Avg Cost Measure for arbitrary distribution. (b) ART simulations with up to 600 K P2P nodes: Range Query Avg Cost Measure for uniform distribution. (c) ART simulations with up to 600 K P2P nodes: Range Query Avg Cost Measure for powlow distribution.

- statistics and management in order to monitor all different strategies natively in the simulator. Statistics include: (i) cost monitoring of importing and retirement of nodes, (ii) the number of failed queries due to node failures and (iii) the number of nodes leading to overlay partition

Self-willing departure of nodes could be simulated following two approaches. It is possible to simulate scenarios that departure of nodes is initiated in parallel, so that random nodes in the network depart simultaneously. A second approach is to set nodes departing in a sequential mode, where each next node departs after another's complete leave. Though the simulator supports both modes, we have seen that sequential departures are not realistic and as a result they tend to hide problems that might appear (e.g., simultaneous departure of a node and its backup node). On the other hand, multiple concurrent departures make P2P protocol designers to deal with such cases.

The classes and packages of the framework include all the necessary code parts to facilitate the researcher in order to detect, control departures and execute simulation using them. The message passing environment is designed to be possible to detect during simulations whether the message recipient is online or not (i.e., it does not take for a fact that the nodes are always available as done in rest of simulators).

Additional details and technical specifications on the failure scenarios related operations are given in the following.

It is common place that sudden departure of nodes without notice can bring large scale problems to routing messages within a P2P network. Such sudden simultaneous departures of multiple nodes makes difficult to failure recovery routing strategies find an alternative path to avoid failure nodes. In such cases, sending

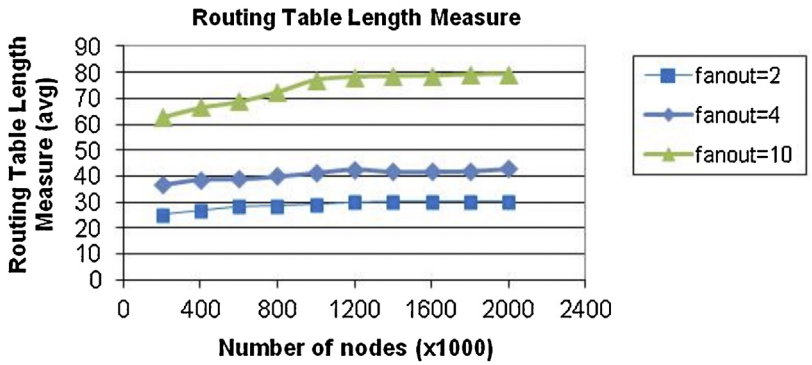

Fig. 13. Baton* Simulations with up to 2 Million P2P nodes: Routing Table Length Measure.

messages to the same node more than one time is probable. In order to overcome such cases, the simulator infrastructure includes tools to store all intermediate nodes that a message visited in its path for the sake of the simulation study even in cases that a protocol does not need such information. As a result, all paths can be stored and studied after each message is sent.

Moreover, to support departure and node failures for any protocol, node selection strategies take into consideration exception lists for nodes that have failed while running any distribution for experiments. Simulator facilities allow passing failed node lists and/or departed node lists to preprocessing of distributions utilized to retrieve randomized node ids during experimental runs.

D-P2P-Sim$^+$ determines the state of each node based on the state of each thread implementing it (RUNNABLE, BLOCKED, TERMINATED). However, in order to handle self-willing node departures and sudden retirement of nodes, the node has to pass through different states during simulation, independent from the thread state that implements it. These states are fixed in the class named

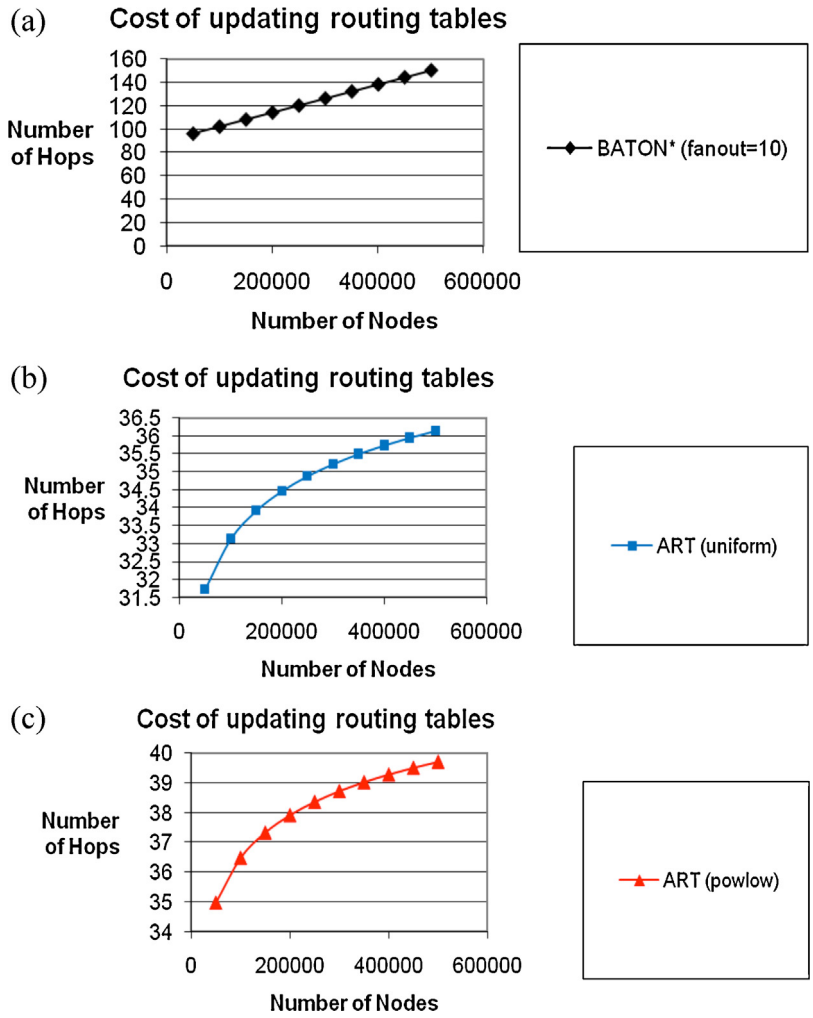

Fig. 14. (a) BATON* Simulations with up to 600 K P2P nodes: Updating Routing Table Avg Cost Measure for arbitrary distribution. (b) ART simulations with up to 600 K P2P nodes: Updating Routing Table Cost Measure for uniform distribution. (c) ART simulations with up to 600 K P2P nodes: Updating Routing Table Cost Measure for powlow distribution.

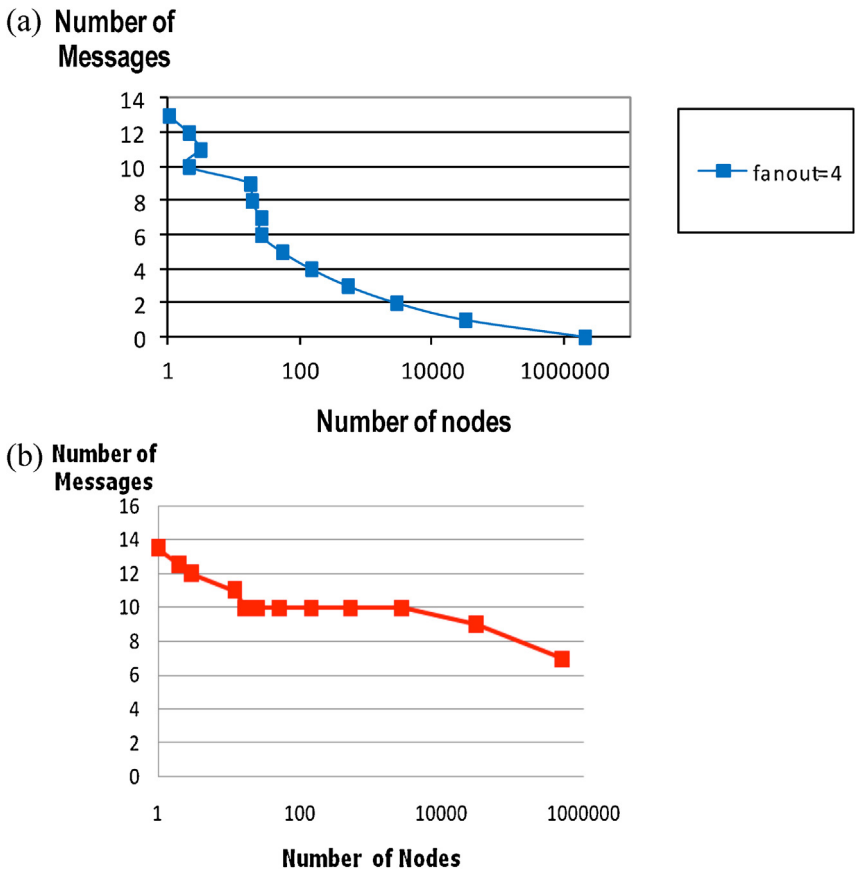

Fig. 15. (a) Baton* Simulations with up to 2 Million P2P nodes: Messages per node measure. (b) ART simulations with up to 1 Million P2P nodes: Messages per node measure.

PeerState that was created as a part of network Overlay package, and they are following: (a) WORKING: online and ready, (b) CANDIDATE_SUBSTITUTE: node substitution by existing node or nodes at the network, (c) VOLUNTARILY_LEFT: node self-willing departure, (d) FAILED: abrupt node failure.

Next, it is crucial to monitor for overlay network partitioning after failure of successive nodes, which results in isolation of nodes. This happens when all routing pointers to nodes outside the isolated partition are broken. In other words, the minimum number of routing pointers that should break in order that a group of nodes is separated from the overlay network equals to S as follows:

$$S = \sum \text{contacts of all nodes of team} - \sum \text{internal contacts between nodes}$$

Furthermore, details on the statistics needed to monitor failures and departures of nodes are: (a) The number of steps (hops) to find the suitable overlay network point for additions in the overlay network. This calculation is realized with the infiltration of message JOIN_RESP from the Network Filter class. (b) Number of steps to find a substitute when a intermediary node wishes to withdraw itself. This calculation is realized with the infiltration of message REPLACEMENT_RESP from the Network Filter class. (c) Number of search queries, additions or deletions of keys that failed, either because the expected node has departed, or because the network has been partitioned and it is not possible to access the requested destination. This calculation is realized with the sum of messages QUERYFAILED_RES from the Network Filter class. In particular, a main target of this framework is to maintain robust collection and analysis of statistical data that results from the experiments at

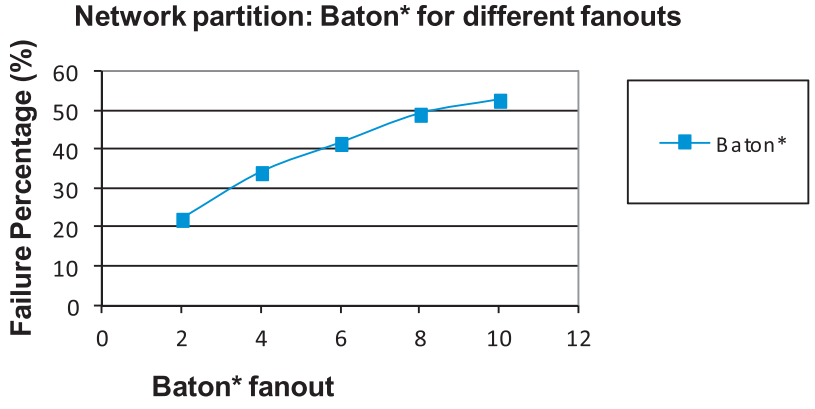

Fig. 16. Baton* Simulations with up to 2 Million P2P nodes: failure percentage measure before network partition.

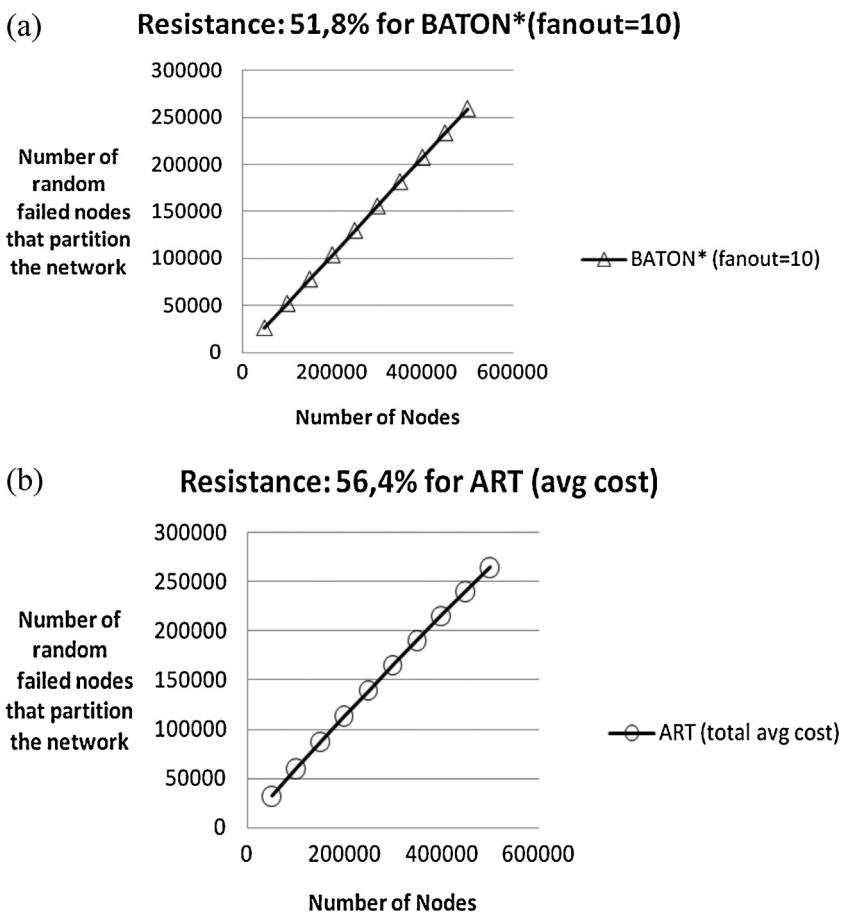

Fig. 17. (a) Baton* Simulations with up to 600 K P2P nodes: Resistance percentage measure. (b) ART simulations with up to 600 K P2P nodes: Resistance percentage avg cost measure.

all new failure detection and overcome scenarios strategies. The importance of experimental analysis is of more major importance, because it confirms the theoretical analysis, it elects problems and potential omissions that had not been located during the theoretical study, and constitutes a proof for the proposal that is presented. Moreover, the more realistic the simulation is, the smaller the divergence of experimental results from the real life results.

7. GUI support

Throughout this section the functional specifications and the Graphical User Interface (GUI) of the D-P2P-Sim$^+$ framework are described. The GUI (see Fig. 9) is organized in such a way that can guide someone easily through the simulation process with no need for any prior knowledge about the simulator.

The GUI includes full management and handling of all the above operations. The GUI gives the user the possibility of setting up simulation experiments for protocols without touching any supplemental files for initial parameters (configuration files) via the appropriate tabs. The tabs of GUI named *Operation* and *Experiments*, respectively are the ones that users implement the operations and setup experiments, respectively. More analytically, by using the tab *Operation*, users can apply self-willing retirement and failures of nodes according to their wishes, within the limits of the overlay network.

Via the tab *Experiments*, the GUI gives the user the option to execute multiple experiments to test search, addition and deletion of keys, so that it evaluates the protocol of his wish. It is also possible to design experiments for mass self-willing departures or failures of nodes. The self-willing retirement may be set to be done in either successive randomized manner or in batch mode where multiple nodes fail together at the same time.

During initiated node failures, one can check if a partitioned group of nodes has been created in the network, using the appropriate button, the so-called *Is the network partitioned*? Respectively, statistical analysis is available at the *Statistics* tab. Following, further details of the experimental procedures are discussed for different setups.

8. Simulation environment

A standalone experimental evaluation at a typical research laboratory computer can be performed either using the GUI or filling out XML configuration files with the necessary the parameters of execution.

First, we demonstrate the efficiency of our framework to deploy a number of distributed protocols and test their lookup performance. Using a single-PC configuration (Intel Core2 Duo CPU @ 3 GHz, 3GB RAM) we simulate 100 K node overlays with *six* different protocols: Chord, BATON*, NBDT, NBDT*, R-NBDT* and ART. Results that register the required time for overlay construction and memory requirements are presented in Fig. 5. Firstly, we note that even a single typical PC can easily host wide scale experiments. Secondly, our framework easily hosts a wide variety of protocols and manages to very efficiently build a large overlay using minimal resources: At most 1 GB of memory is required for 100 K overlay construction and full functionality. Execution times for this mode

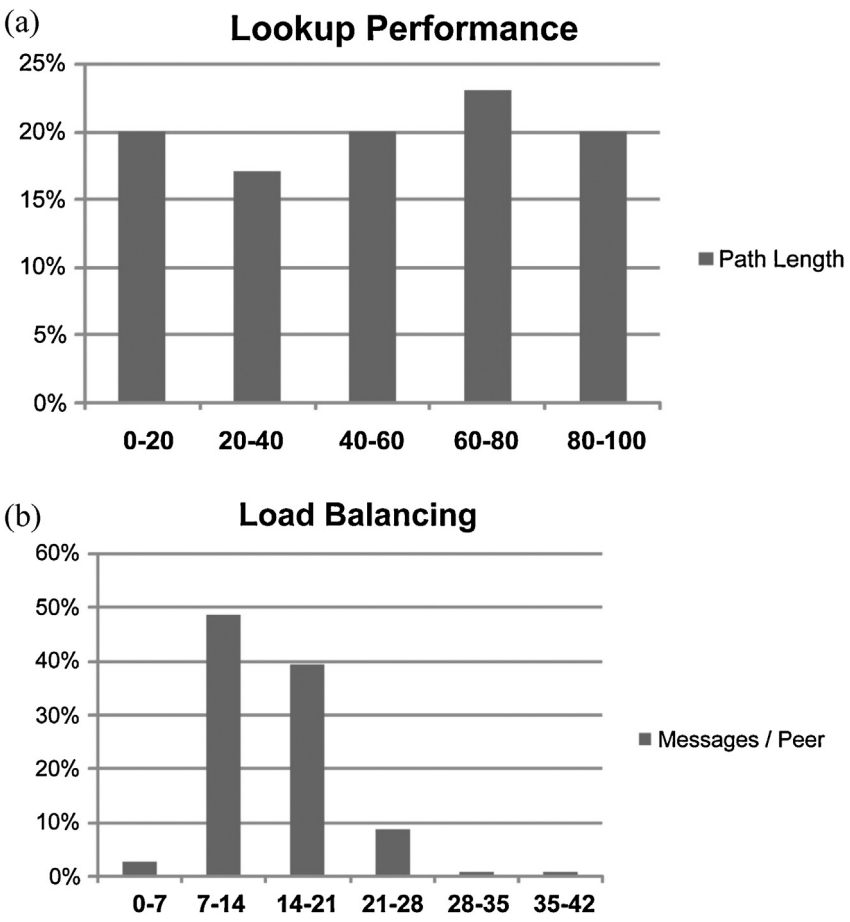

Fig. 18. (a) More metrics for Chord path-length: Chord 10 K network. (b) More metrics for ART: Load Balancing (msg/node) Cost Measure for ART 10 K network.

are also very small, ranging from 35 min to at most 115 min for NBDT.

The distributed environment (see Fig. 10) that was used for the remainder of the experiments consists of a total of 5 PCs (Intel(R) Xeon(R) CPU E5504 @ 2.00 GHz, 8BG RAM) running over Ubuntu 9.10. To show the potential of the framework we evaluated the performance of a number of protocols. Exploiting the efficiency and its ability to function in a distributed environment, we executed large-scale simulations, altering the size of the network from 100,000 up to 2,000,000+ nodes with different flavors of protocols.

Node failure and recovery scenarios have been evaluated as well as node departures and substitutions to test fault tolerance per protocol. For all this spectrum of nodes and fanouts the configurations used, took over 8 h to deliver full output. Trials have been made with single, and 2-core up to 8-core processors and with 0.5 GB up to 16 GB of RAM memory verified that the simulator scaled without problems and did not need any changes of its configuration. This is also verified for experiments with 1 up to 12 PCs which is a number of PCs expected available at a typical University Lab (WAN experiments are also done on 50 Planetlab points, please see section below). Furthermore the number of nodes simulated scaled near-linearly.

8.1. D-P2P-Sim$^+$ experimental evaluation

Figs. 11 and 12 present results for exact-match (or lookup) and range queries, respectively, while experimenting with two (2) well known protocols: Baton* for up to two million nodes and ART for up to 600 K nodes. The performance of P2P protocols is greatly dependent on the average path length between two random network nodes. Fig. 10 shows the path used in order to define the exact-match node-positions where insertions and deletion of keys have to be executed. Results verify that query cost for Baton* and ART is logarithmically and sub-logarithmically increasing, respectively, with the network population. Moreover, experiments show that query costs are decreasing with a fanout increase. In particular, the gain received from the fanout increase is larger when the network becomes more massive. This is expected as the larger the number of nodes, the smaller the tree height becomes with the fanout increase (in terms of rate). As a consequence, we have verified for overlays up to 2 M nodes that the cost of search, insert and delete in Baton* and ART protocols is $O(\log_m N)$ and $O(\log_b{}^2 \log N)$, respectively, where m is the fanout factor in Baton*, b is the fanout in ART and N is the number of nodes. Note that, according to Baton* p2p structure, original results presented in (Jagadish et al., 2006) are up to 10 K nodes.

Fig. 13 shows the average routing table length for Baton*. It is clear that with an increase in the network size, the routing information that has to be maintained per node is also increasing: Increasing the number of nodes, more levels are created in the tree and therefore, the number of neighbors maintained at each node is increasing too. Moreover, a fanout increase leads to an increase of routing table length and, as a result, the cost of updating it. Fig. 14 shows cost measurements for Updating Routing Tables in Baton* and ART structures for a variety of input distributions. Experiments verify that faster search is possible using larger routing tables.

Fig. 15 presents the load that each node receives, counting the number of messages received for any of the operations for node populations up to 2 M nodes and for 3 K operations. As shown, in both Baton* and ART structures, the maximum number of messages

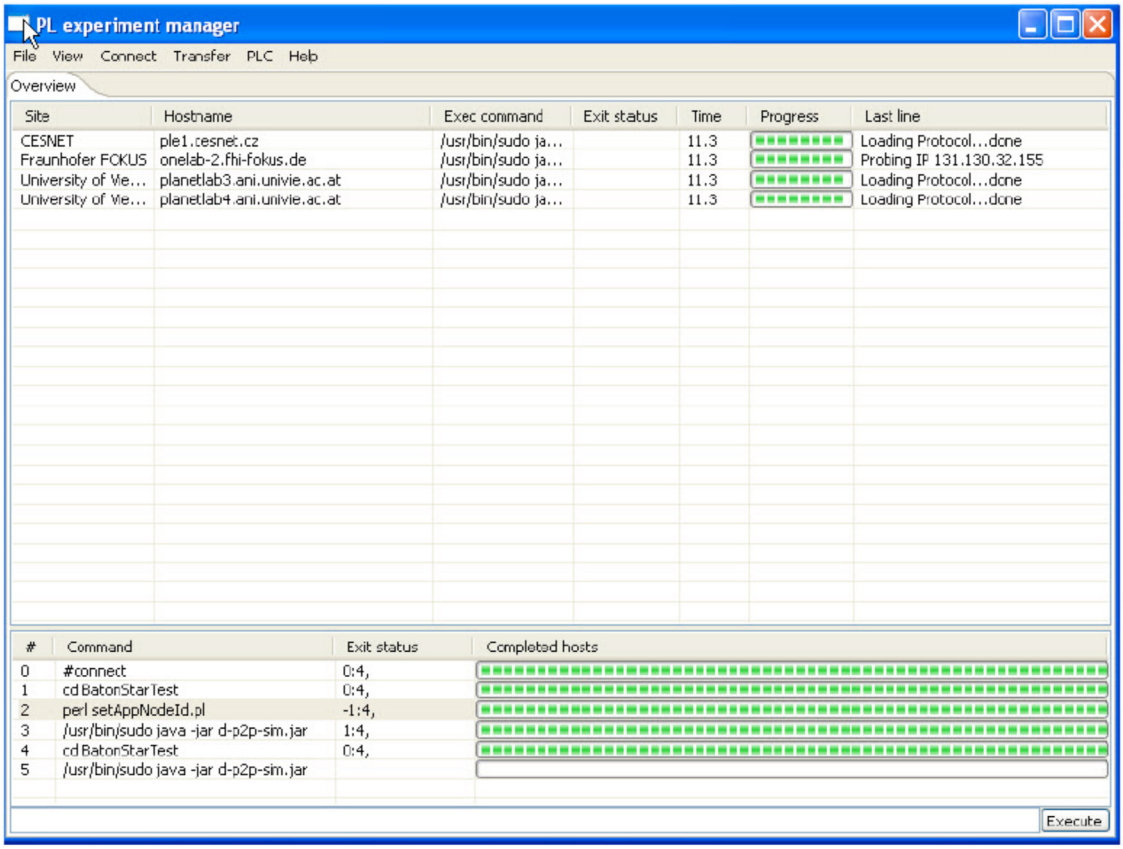

Fig. 19. Executing D-P2P-Sim+ on PlanetLab Network.

in the experiments reached 13 and this only happens for a single node. On the contrary, 30,590 nodes receive a single message in Baton* structure and ~1 M nodes receive 6 messages in ART structure. As a result, we note that both protocols successively balance the load among nodes. This also verifies that D-P2P-Sim$^+$ includes all the necessary features and tools to detect and research on load balancing techniques for P2P network protocols. As discussed in previous section, a list of collectable metrics is available to evaluate and experiment with protocols at application, protocol and system level.

We also show that our framework is able to collect connection-specific metrics that can be used to evaluate node-failure and recovery strategies on networks with multi million nodes. The problem that massive failures cause is the invalidation of links among them. As the search procedures have to overcome non-reachable connections, it is hard to choose a path that does not include failed nodes. Queries oscillate inside the overlay until the alternative path is determined. Thus an increase in node failures is expected to result in an increase to search costs. In our setup, a network of 1 M nodes is initialized, with a 10% of randomly chosen nodes being assigned to leave abruptly (sudden node death), without rebuilding the network. The experiments continue increasing the percentage of nodes failing in the network by 1% at each round. In each step, the network is checked in order to verify that it is not partitioned into isolated areas that cannot communicate. All the experiments are repeated for fanout 2 up to 10 for Baton*.

Fig. 16 shows the average number of nodes that is expected to fail before the network is partitioned. The results verify that the network is resistant to failures when a quarter of the nodes fails for *fanout* = *2*. A fanout increase results in higher degrees of tolerance. Fig. 17 shows the resistance percentage measure for Baton* (fanout = 10) and ART, respectively. For 50.000 nodes, BATON* and ART resist till 25.900 and 232.031 nodes have randomly failed, respectively. For 500.000 nodes, BATON* and ART resist until 259.000 and 264.109 nodes have randomly failed, respectively. Thus, the average resistance is 51.8% for BATON* (fanout = 10) and almost 56.4% for ART. The existence of extra (LSI and CI) routing tables in ART result in higher degrees of tolerance, and as a result ART is more Robust than BATON*.

Fig. 18 shows a number of more metrics for Chord, and ART. Fig. 18(a) verifies the logarithmic behavior of Chord protocol as well as Fig. 18(b) verifies that even ART retains large CI tables at each level and the probability of hot-spots existence is higher than in Chord and BATON*, it still successively balances the visiting rate among nodes.

Next, we will describe more metrics for a variety of p2p protocols at the PlanetLab.

8.2. D-P2P-Sim$^+$ at the PlanetLab

PlanetLab is a distributed research networks infrastructure comprising 1091 computers, which are found in 505 different research locations. Each network participant allocates computers and is given the possibility to use resources from all the network to implement large scale experiments. The need for transparent transfer of experimental configuration and evaluation into large-scale infrastructures with limited changes is a common vision for researchers and research developers. D-P2P-Sim$^+$ has been verified to be easy to setup and execute at the PlanetLab network using the same experimental configuration as that inside a research lab. In order to avoid involvement of PlanetLab computers that are out of reach or overloaded the CoMon tool can be utilized that it provides statistics for PlanetLab available resources, both at nodes and slice level.

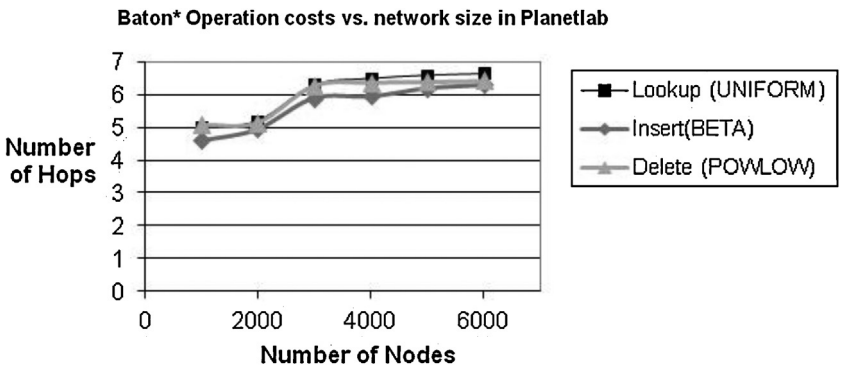

Fig. 20. More metrics for BATON*: Baton* Operation costs vs. network size on Planetlab.

In order to test and experiment on a large number of computers, a tool that will connect and execute command in parallel is needed. For this PlanetLab *tools of management slice* are employed (Fig. 19). In the experimental simulation scenarios, 50 PlanetLab points were selected, based on their statistics according to the CoMon tool, so that they would not face severe problems of network interconnection. In each evaluation, node failure and recovery scenarios have been evaluated as well as node departures and substitution in order to test fault tolerance and robustness of each protocol tested. During experimentation at the PlanetLab, we observed that execution times are an order of magnitude larger than those required in our lab.

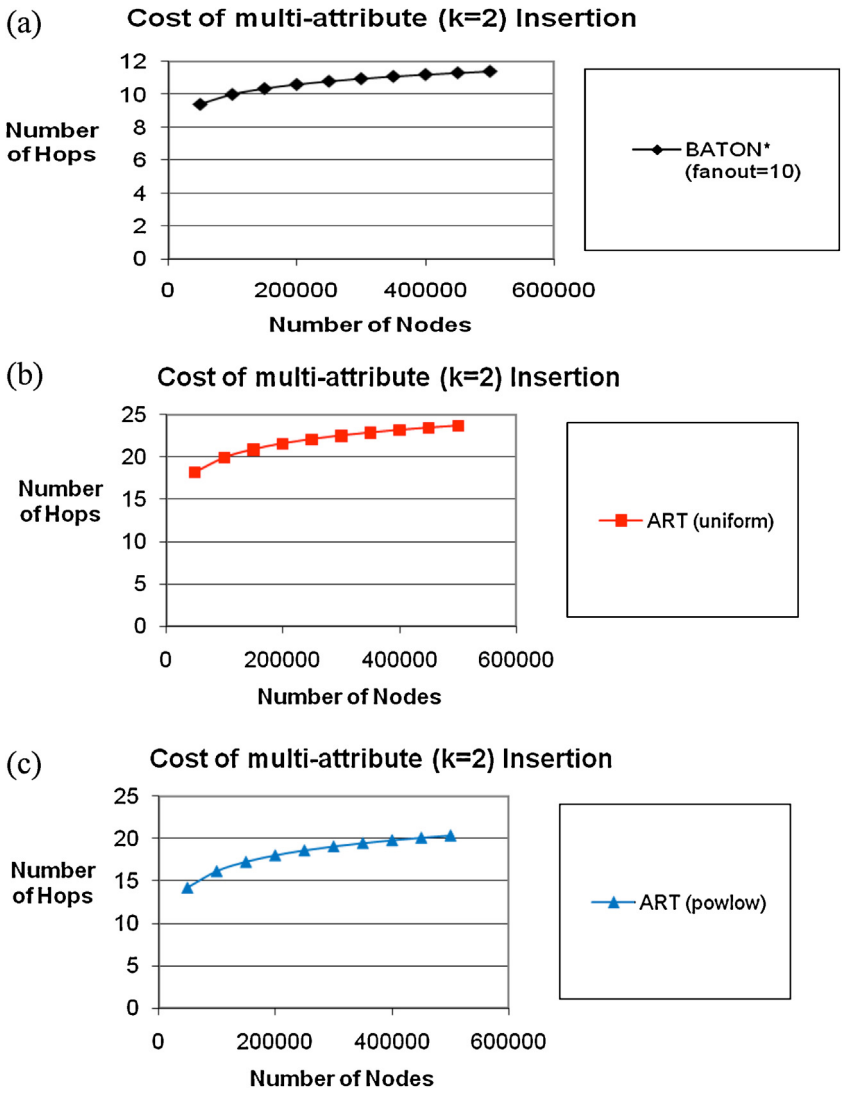

Fig. 21. (a) More metrics for BATON* with up to 600 K P2P nodes: 2-Dimensional Insertion Cost Measure on Planetlab (Avg Cost in a variety of random distributions). (b) More metrics for ART with up to 600 K P2P nodes: 2-Dimensional Insertion Cost Measure on Planetlab for uniform distribution. (c) More metrics for ART with up to 600 K P2P nodes: 2-Dimensional Insertion Cost Measure on Planetlab for powlow distribution.

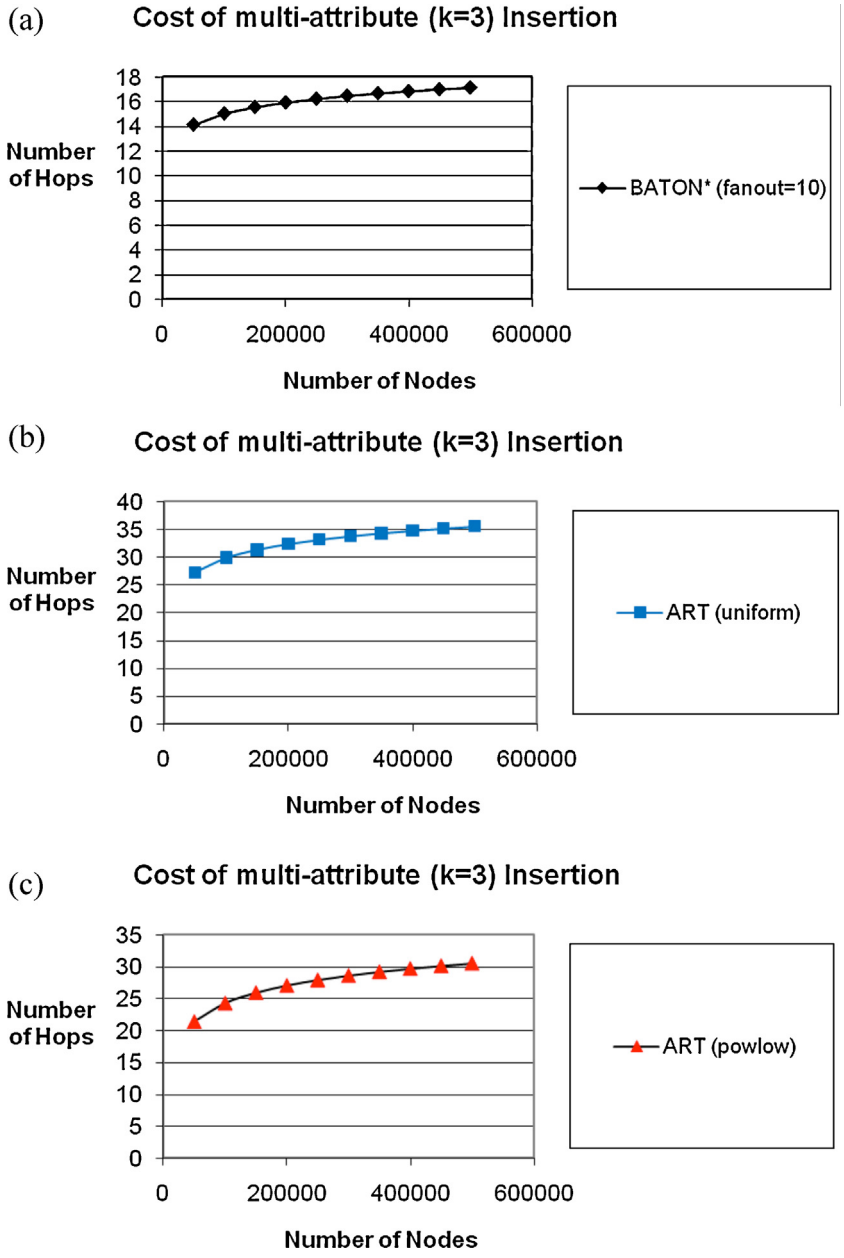

Fig. 22. (a) More metrics for BATON* with up to 600 K P2P nodes: 3-Dimensional Insertion Cost Measure on Planetlab (Avg Cost in a variety of random distributions). (b) More metrics for ART with up to 600 K P2P nodes: 3-Dimensional Insertion Cost Measure on Planetlab for uniform distribution. (c) More metrics for ART with up to 600 K P2P nodes: 3-Dimensional Insertion Cost Measure on Planetlab for powlow distribution.

There exist cases where 6000 P2P nodes in PlanetLab take approximately 7.5 h to return results. Slices of the PlanetLab are often over-loaded and large-scale experiments face enormous delays. Moreover, network communication among PlanetLab slices is an additional overhead that has to be taken into consideration when planning experiments. However, Planetlab is useful in order to verify that an engine under research is possible to work even under very large communication obstacles and most importantly in wide area networks. In Fig. 19, one notices how D-P2P-Sim$^+$ is executed progressively for Baton* P2P at the PlanetLab nodes.

Fig. 20 depicts cost measurements for a variety of operations (search, insert, delete) each of which is executed at the PlanetLab for a variety of input distributions. The logarithmic cost of BATON* protocol for lookup and update operations was verified again.

Figs. from 21 up to 22 depict the insertion cost measurements in PlanetLab for Baton* and ART, in multi-attribute case, where we have k $(2 \le k \le 3)$ separate indexes. In particular, BATON* requires $k \log N$ hops and ART requires $k \log 2\text{blog}(N/\text{polylog}\, N) + k \log(\text{polylog}\, N)$ hops. For arbitrary k $(2 \le k \le 3)$ and b = 2, BATON*is always 2 times faster (see figures from 21 up to 22).

Finally, the results are analogous for multi-attribute exact-match (see Fig. 23) and range queries (see Fig. 24), respectively. We used k separate indexes for running the k separate queries in parallel, where for $k \le 6$ the join processing overhead is negligible. The later does not hold for $k > 6$. Thus, for $k \le 6$ and $b = 2$, BATON* is almost 2 times faster for both multiattribute exact-match and range queries, respectively (see Figs. 23 and 24).

8.3. Accuracy aspects of D-P2P-Sim+

According to (Basu et al., 2013), if P2P simulation is able to accurately simulate real P2P networks, it can give the ability to design, implement and evaluate new technologies in a controlled

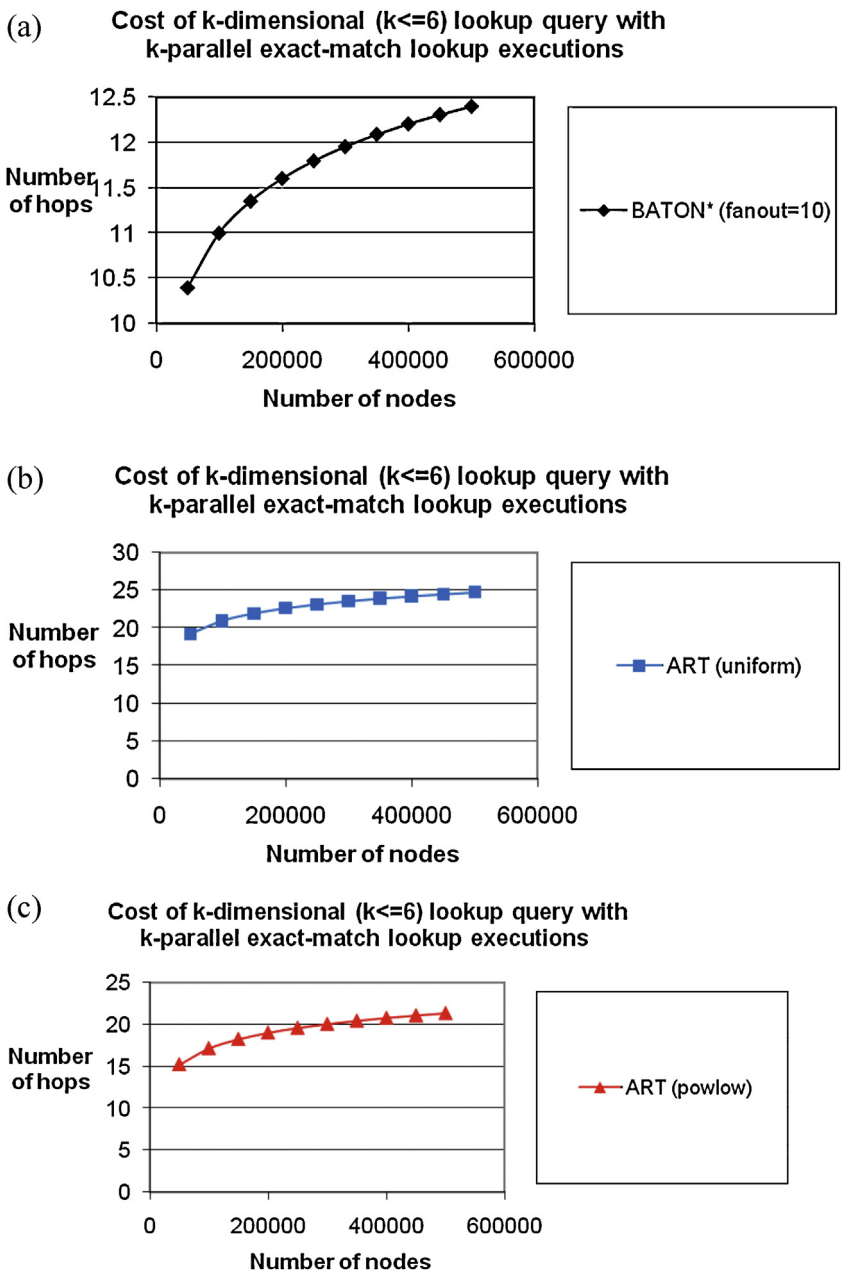

Fig. 23. (a) More metrics for BATON* with up to 600 K P2P nodes: Multi-Dimensional Lookup Cost Measure on Planetlab (Avg Cost in a variety of random distributions). (b) More metrics for ART with up to 600 K P2P nodes: Multi-Dimensional Lookup Cost Measure on Planetlab for uniform distribution. (c) More metrics for ART with up to 600 K P2P nodes: Multi-Dimensional Lookup Cost Measure on Planetlab for powlow distribution.

environment, while allowing for the collection of statistics with greater ease, compared to deployment with real users. In general, it is practically impossible to capture the full complexity of the Internet in any controlled environment. To achieve even higher accuracy when using the proposed simulator, we have designed, deployed, experimented and verified the results presented above over the top real world test bed environment known as the global research network PlanetLab (https://www.planet-lab.org). PlanetLab test bed consists of over 1000 computers situated at participating sites across the globe. PlanetLab has already several years' history and it continuously expands with new Universities and Research Centers joining its infrastructure.

However, while such an infrastructure provides valuable real-world data in terms of network latency, the disadvantage is that a researcher has to depend on the machines with limited and sometimes unexpected availability and to make appropriate configuration steps each time on PlanetLab in order to reproduce results. This can be a tedious task since the machines belong in different administrative domains, each with its own support staff and hardware/software upgrade schedule. Therefore, tracking issues and problems across such a large number of real computers is a challenging and daunting task. To avoid all this overhead and allow the researcher to focus on experimental measurements, the proposed solution provides a useful approach-strategy to smooth and overcome difficulties.

First, D-P2P-Sim+ enables the researcher to perform real-world simulation using multiple nodes on small and medium numbers of physical nodes. Physical nodes in this case are computers that the researcher may have available locally in a laboratory or at collaborating laboratories. Having simulation in such laboratory environments allow the researcher to schedule more easily and guarantee their availability for simulation purposes than in the

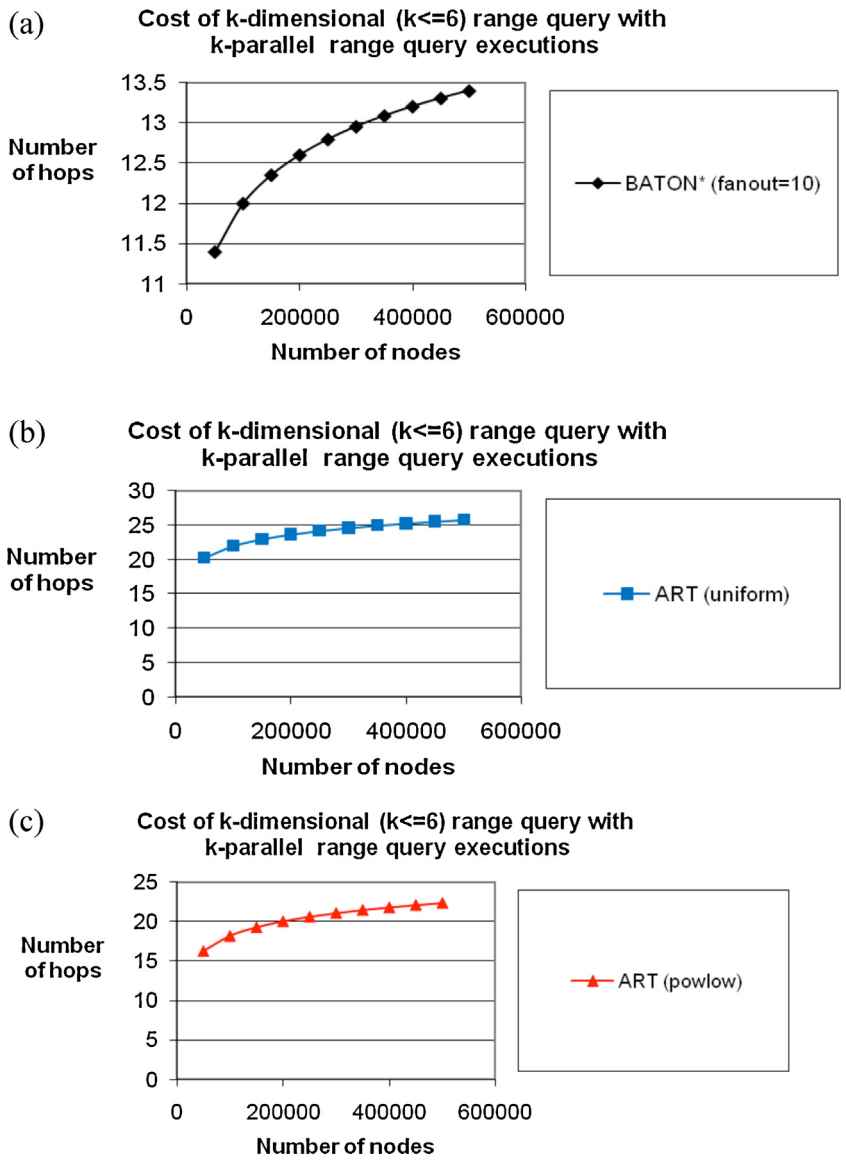

Fig. 24. (a) More metrics for BATON* with up to 600 K P2P nodes: Multi-Dimensional Range Query Cost Measure on Planetlab (Avg Cost in a variety of random distributions). (b) More metrics for ART with up to 600 K P2P nodes: Multi-Dimensional Range Query Cost Measure on Planetlab for uniform distribution. (c) More metrics for ART with up to 600 K P2P nodes: Multi-Dimensional Range Query Cost Measure on Planetlab for powlow distribution.

large scale network of PlanetLab. A key feature of D-P2P-Sim+ is that it may simulate multiple P2P nodes at each physical node used in local or wide area networks. Therefore, the researcher is able to setup simulation experiments with large numbers of nodes P2P networks in his/her laboratory efficiently, before deployment and exposure to the vagaries of the real world.

D-P2P-Sim+ allows for the representation of nodes (which simulate peers) with incoming and outgoing message queues. Over time steps these message queues are processed, passing messages between peer outgoing and incoming queues, simulating the communication in a P2P network. The time step length is possible to be configured accordingly at each node separately or evenly across all nodes in the experimental setup in order depict appropriate end-to-end delays as determined by real life network topologies and prevailing network patterns (Lin et al., 2005). All nodes are aware of their neighbors, and therefore send and receive messages to their neighbors via their message queues (Vahdat et al., 2002). Moreover, D-P2P-Sim+ is possible to simulate node failure and recovery strategies and experimental scenarios to evaluate the behavior of the P2P algorithm under research in such cases.

Next, the novel D-P2P-Sim+ is ready to be used through the PlanetLab network, whenever the researcher needs additional physical nodes to experiment with a simulation for example to insert additional nodes in the P2P of to utilize and test a specific network topology of previously unknown characteristics. Please note that, we have verified that experimental results from PlanetLab about P2P networks do follow the produced results at a single laboratory and multiple collaborating laboratories as shown at the figures above. We show that a limited number of modern typical PCs can accurately simulate network latency and failures of multi-millions of nodes in a laboratory environment.

Based on the above principles, we presented the novel D-P2P-Sim+, which has the ability to be appropriately configured in both, small-scale LAN network laboratories and if needed in large scale a real world test bed environment such as PlanetLab. Evaluation has shown that PlanetLab experimental results hopefully verified a "beautiful" reproduction of local lab results with a limited number of PCs only. Of course, there is an observed difference in effectiveness due to an expected statistical error between the real and the theoretical protocol performance at each testbed. For example,

Table 2
"Lookup Query" simulator of 50 Planet Lab-nodes with data drawn by UNIFORM distribution and the expected theoretical evaluation $O(\log_m N)$ for BATON* protocol (with constant parameter $c = 1$ and fun-out $m = 4$).

Number of simulated Nodes@**50 PlanetLab Nodes** testbed	Number of hops for **LOOKUP** data drawn by **UNIFORM** distribution	The expected theoretical performance
1000	5	4.982892
2000	5.3	5.482892
3000	6.1	5.775373
4000	6.4	5.982892
5000	6.5	6.143856
6000	6.7	6.275373

Table 3
"Lookup Query" simulator of 5 LAN-nodes with data drawn by UNIFORM distribution and the expected theoretical evaluation $O(\log_m N)$ for BATON* protocol (with constant parameter $c = 1$ and fun-out $m = 4$).

Number of simulated Nodes@**5 LAN-nodes** testbed	Number of hops for **LOOKUP** data drawn by **UNIFORM** distribution	The expected theoretical performance
200,000	10	8.80482
400,000	11	9.30482
600,000	12	9.597301
800,000	12.1	9.80482
1,000,000	11.9	9.965784
1,200,000	12	10.0973
1,400,000	12.4	10.2085
1,600,000	13	10.30482
1,800,000	13.9	10.38978
2,000,000	13.8	10.46578

running the BATON* (the most famous) protocol at each testbed (with 50 PlanetLab-nodes and 5 LAN-nodes, respectively) we provide the following statistical test analysis for a set of 1000 queries (lookup, insert, delete, join/leave) and a variety of smooth data distributions (uniform, beta and powlow).

In Table 2, the average number of real and theoretical hops is 6 and 5.773, respectively. Thus, the average observed difference is 0.226 or the expected statistical error is 3.76%.

In Table 3, the average number of real and theoretical hops is 12.21 and 9.894, respectively. Thus, the average observed difference is 2.315 or the expected statistical error is 18.9%.

In Table 4, the average number of real and theoretical hops is 5.675 and 5.773, respectively. Thus, the average observed difference is 0.098 or the expected statistical error is 1.74%.

In Table 5, the average number of real and theoretical hops is 11.63 and 9.894, respectively. Thus, the average observed difference is 1.735 or the expected statistical error is 14.9%.

In Table 6, the average number of real and theoretical hops is 5.9 and 5.77, respectively. Thus, the average observed difference is 0.126 or the expected statistical error is 2.13%.

Table 4
"INSERT Query" simulator of 50 Planet Lab-nodes with data drawn by BETA distribution and the expected theoretical evaluation $O(\log_m N)$ for BATON* protocol (with constant parameter $c = 1$ and fun-out $m = 4$).

Number of simulated Nodes@**50 PlanetLab Nodes** testbed	Number of hops for **INSERTING** data drawn by **BETA** distribution	The expected theoretical performance
1000	4.7	4.982892
2000	4.9	5.482892
3000	5.85	5.775373
4000	6	5.982892
5000	6.2	6.143856
6000	6.4	6.275373

Table 5
"INSERT Query" simulator of 5 LAN-nodes with data drawn by BETA distribution and the expected theoretical evaluation $O(\log_m N)$ for BATON* protocol (with constant parameter $c = 1$ and fun-out $m = 4$).

Number of simulated Nodes@**5 LAN-nodes** testbed	Number of hops for **INSERTING** data drawn by **BETA** distribution	The expected theoretical performance
200,000	9.9	8.80482
400,000	10.4	9.30482
600,000	11.5	9.597301
800,000	11.9	9.80482
1,000,000	11.2	9.965784
1,200,000	11.3	10.0973
1,400,000	11.9	10.2085
1,600,000	12.5	10.30482
1,800,000	12.9	10.38978
2,000,000	12.8	10.46578

Table 6
"DELETE Query" simulator of 50 Planet Lab-nodes with data drawn by POWLOW distribution and the expected theoretical evaluation $O(\log_m N)$ for BATON* protocol (with constant parameter $c = 1$ and fun-out $m = 4$).

Number of simulated Nodes@**50 PlanetLab Nodes** testbed	Number of hops for **DELETING** data drawn by **POW-LOW** distribution	The expected theoretical performance
1000	5	4.982892
2000	5.2	5.482892
3000	6.2	5.775373
4000	6.2	5.982892
5000	6.35	6.143856
6000	6.45	6.275373

In Table 7, the average number of real and theoretical hops is 11.99 and 9.89, respectively. Thus, the average observed difference is 2.09 or the expected statistical error is 17.4%.

In Table 8, the average number of real and theoretical hops is 23.6 and 23.09, respectively. Thus, the average observed difference is 0.504 or the expected statistical error is 2.13%.

In Table 9, the average number of real and theoretical hops is 47.96 and 39.57, respectively. Thus, the average observed difference is 8.38 or the expected statistical error is 17.4%.

In Table 10, the average number of real and theoretical hops is 22.7 and 23.09, respectively. Thus, the average observed difference is 0.395 or the expected statistical error is 1.74%.

In Table 11, the average number of real and theoretical hops is 46.52 and 39.57, respectively. Thus, the average observed difference is 6.94 or the expected statistical error is 14.9%.

The detailed statistical analysis previously presented proves that our simulator does capture real-world performance

Table 7
"DELETE Query" simulator of 5 LAN-nodes with data drawn by POW-LOW distribution and the expected theoretical evaluation $O(\log_m N)$ for BATON* protocol (with constant parameter $c = 1$ and fun-out $m = 4$).

Number of simulated Nodes@**5 LAN-nodes** testbed	Number of hops for **DELETING** data drawn by **POW-LOW** distribution	The expected theoretical performance
200,000	9.9	8.80482
400,000	10.8	9.30482
600,000	11.9	9.597301
800,000	12	9.80482
1,000,000	11.8	9.965784
1,200,000	11.5	10.0973
1,400,000	12.5	10.2085
1,600,000	12.9	10.30482
1,800,000	13.5	10.38978
2,000,000	13.1	10.46578

Table 8
"Leave/Failure-Node Query" simulator of 50 Planet Lab-nodes after POWLOW leaves/failures of nodes and the expected theoretical evaluation $O(m \log_m N)$ for BATON* protocol (with constant parameter $c = 1$ and fun-out $m = 4$).

Number of simulated Nodes@**50 PlanetLab Nodes** testbed	Number of hops for **UPDATING ROUTING-TABLES** after **POW-LOW** leaves/failures of nodes	The expected theoretical performance
1000	20	19.93157
2000	20.8	21.93157
3000	24.8	23.10149
4000	24.8	23.93157
5000	25.4	24.57542
6000	25.8	25.10149

Table 9
"Leave/Failure-Node Query" simulator of 5 LAN-nodes after POWLOW leaves/failures of nodes and the expected theoretical evaluation $O(m \log_m N)$ for BATON* protocol (with constant parameter $c = 1$ and fun-out $m = 4$).

Number of simulated Nodes@**5 LAN-nodes** testbed	Number of hops for **UPDATING ROUTING-TABLES** after **POW-LOW** leaves/failures of nodes	The expected theoretical performance
200,000	39.6	35.21928
400,000	43.2	37.21928
600,000	47.6	38.38921
800,000	48	39.21928
1,000,000	47.2	39.86314
1,200,000	46	40.38921
1,400,000	50	40.83399
1,600,000	51.6	41.21928
1,800,000	54	41.55913
2,000,000	52.4	41.86314

Table 10
"Join_Node Query" simulator of 50 Planet Lab-nodes after **BETA** Joins of nodes and the expected theoretical evaluation $O(m \log_m N)$ for BATON* protocol (with constant parameter $c = 1$ and fun-out $m = 4$).

Number of simulated Nodes@**50 PlanetLab Nodes** testbed	Number of hops for **UPDATING ROUTING-TABLES** after **BETA** Joins of nodes	The expected theoretical performance
1000	18.8	19.93157
2000	19.6	21.93157
3000	23.4	23.10149
4000	24	23.93157
5000	24.8	24.57542
6000	25.6	25.10149

Table 11
"Join_Node" Query simulator of 5 LAN-nodes after **BETA** Joins of nodes and the expected theoretical evaluation $O(m \log_m N)$ for BATON* protocol (with constant parameter $c = 1$ and fun-out $m = 4$).

Number of simulated Nodes@**5 LAN-nodes** testbed	Number of hops for **UPDATING ROUTING-TABLES** after **BETA** Joins of nodes	The expected theoretical performance
200,000	39.6	35.21928
400,000	41.6	37.21928
600,000	46	38.38921
800,000	47.6	39.21928
1,000,000	44.8	39.86314
1,200,000	45.2	40.38921
1,400,000	47.6	40.83399
1,600,000	50	41.21928
1,800,000	51.6	41.55913
2,000,000	51.2	41.86314

characteristics accurately with appropriate configuration. In particular, it proves that the expected statistical error in PlanetLab testbed is enough smaller than in 5 LAN-Nodes testbed, since the former testbed is based on the statistical performance offered by CoMon tool, which guarantees neither network problems nor luck of network resources.

Overall, we have designed D-P2P-Sim+ to include: (i) features to allow reducing the complexity of the simulated topology, (ii) options to allow multiple virtual nodes operating onto a single physical machine, and (iii) introducing synthetic background traffic to dynamically change network characteristics and to simulate network failures.

9. Conclusions

D-P2P-Sim$^+$ is presented as a novel framework to support simulation with multi million nodes storing large amount of data collected in a variety of IoT and Web 2.0 applications. It is an ideal platform for processing the collected data in a fully customized way. It includes a robust framework to work easily on failure and recovery scenarios. All operations as well as robust statistics are available through a simple java GUI. Experimental evaluation in a real world test bed, such as PlanetLab, verified accurately the theoretic analysis of a variety of p2p protocols for up to 2 M nodes. Additional technical details, architectural blueprints and multiple snapshots are available online at the framework's web-page.

Acknowledgements

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) – Research Funding Program: Thales. Investing in knowledge society through the European Social Fund.

This research has been co-financed by the European Union (European Social Fund – ESF) and Greek national funds through the Operational Program "Education and Lifelong Learning" of the National Strategic Reference Framework (NSRF) – Research Funding Program: Cooperation 2011 Partnerships of Production and Research.

We would like also to thank Prof. E. Kafeza for helpful comments and fruitful discussions.

References

2011. A. S. Report on Skype Usage, Retrieved from: http://news.softpedia.com/news/skype usage-surges-as-27-million-people-chat-simultaneously-177523.shtml

Aspnes, J., Shah, G., 2007. Skip graphs. ACM Trans. Algorithms 3 (4), Article No. 37.

Atzori, L., Iera, A., Morabito, G., 2010. The Internet of things: a survey. Comput. Netw. 54 (15), 2787–2805.

Basu, A., Fleming, S., Stanier, J., Naicken, S., Wakeman, I., Gurbani, V.K., 2013. The state of peer-to-peer network simulators. ACM Comput. Surv. 45 (4), Article No. 46.

Bertino, E., Guerrini, G., Mesiti, M., 2004. A matching algorithm for measuring the structural similarity between an XML document and a DTD and its applications? Inf. Syst. 29 (1), 23–46.

Bobelin, L., Legrand, A., González Márquez, D., Navarro, P., Quinson, M., Suter, F., Thiery, C., 2012. Scalable multi-purpose network representation for large scale distributed system simulation. In: Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Ottawa, Canada, pp. 220–227.

Brusilovsky, P., Maybury, T., 2002. From adaptive hypermedia to the adaptive web. Commun. ACM 45 (5), 31–33.

Brusilovsky, P., Kobsa, A., Nejdl, W. (Eds.), 2007. The Adaptive Web: Methods and Strategies of Web Personalization, LNCS vol. 4321. Springer.

Casanova, H., Legrand, A., Quinson, M., 2008. SimGrid: a generic framework for large-scale distributed experiments. In: Proceedings of the 10th International Conference on Computer Modeling and Simulation (UKSIM), Cambridge, UK, pp. 126–131.
Chen, L., Cui, B., Lu, H., 2011. Constrained skyline query processing against distributed data sites. IEEE Trans. Knowl. Data Eng. 23 (2), 204–217.
Cowie, J., Liu, H., Liu, J., Nicol, D., Ogielski, A., 1999. Towards realistic million-node Internet simulation. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA), Las Vegas, NV, pp. 2129–2135.
Cuzzocrea, A., 2006. Combining multidimensional user models and knowledge representation and management techniques for making web services knowledge-aware. Web Intell. Agent Syst. 4 (3), 289–312.
Jagadish, H.V., Ooi, B.C., Tan, K.-L., Vu, Q.H., Zhang, R., 2006. Speeding up search in peer-to-peer networks with a multi-way tree structure. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, Chicago, IL, pp. 1–12.
Lai, K.-C., Yu, Y.-F., 2012. A scalable multi-attribute hybrid overlay for range queries on the cloud. Inf. Syst. Front. 14 (4), 895–908.
Lin, S., Pan, A., Guo, R., Zhang, Z., 2005. Simulating large-scale P2P systems with the WiDS toolkit. In: Proceedings of the 13th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS), Atlanta, DA, pp. 415–424.
Mayer, T., Coquil, D., Schoernich, C., Kosch, H., 2012. RCourse: a robustness benchmarking suite for publish/subscribe overlay simulations with Peersim. In: Proceedings of the 1st Workshop on P2P and Dependability (P2P-Dep), Sibiu, Romania, Article No. 3.
Montresor, A., Jelasity, M., 2009. PeerSim: a scalable P2P simulator. In: Proceedings of the 9th International Conference on Peer-to-Peer Computing (P2P), Seattle, WA, pp. 99–100.
Miorandi, D., Sicari, S., De Pellegrini, F., Chlamtac, I., 2012. Internet of things: vision, applications and research challenges. Ad Hoc Netw. 10 (7), 1497–1516.
Naicken, S., Livingston, B., Basu, A., Rodhetbhai, S., Wakeman, I., Chalmers, D., 2007. The state of peer-to-peer simulators and simulations. SIGCOMM Comput. Commun. Rev. 37 (2), 95–98.
Personalization Reports (n.d.). Retrieved from: http://www.choicestream.com/who/news.php
Pugh, W., 1990. Skip lists: a probabilistic alternative to balanced trees. Commun. ACM 33 (6), 668–676.
Quinson, M., Rosa, C., Thiery, C., 2012. Parallel simulation of peer-to-peer systems. In: Proceedings of the 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), Ottawa, Canada, pp. 668–675.
Shmueli-Scheuer, M., Roitman, H., Carmel, D., Mass, Y., Konopnicki, D., 2010. Extracting user profiles from large scale data. In: Proceedings of the Workshop on Massive Data Analytics on the Cloud (MDAC), Raleigh, NC, Article No. 4.
Sioutas, S., 2008. NBDT: an efficient p2p indexing scheme for web service discovery. J. Web Eng. Technol. 4 (1), 95–113.
Sioutas, S., Papaloukopoulos, G., Sakkopoulos, E., Tsichlas, K., Manolopoulos, Y., 2009. A novel distributed p2p simulator architecture: D-p2p-sim. In: Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM), Hong Kong, China, pp. 2069–2070.
Sioutas, S., Triantafillou, P., Papaloukopoulos, G., Sakkopoulos, E., Tsichlas, K., Manolopoulos, Y., 2013. ART: Sub-logarithmic decentralized range query processing with probabilistic guarantees. Distrib. Parallel Databases 31 (1), 71–109.
Teng, H.Y., Lin, C.N., Hwang, R.-H., 2014. A self-similar super-peer overlay construction scheme for super large-scale P2P applications. Inf. Syst. Front. 16 (1), 45–58.
Vahdat, A., Yocum, K., Walsh, K., Mahadevan, P., Kostic, D., Chase, J., Becker, D., 2002. Scalability and accuracy in a large-scale network emulator. In: Proceedings of the 5th Symposium on Operating Systems Design and Implementation (OSDI), Boston, MA, pp. 271–284.
Wang, J., Sharman, R., Ramesh, R., 2008. Shared content management in replicated web systems: a design framework using problem decomposition. IEEE Trans. Syst. Man Cybern. C 38 (1), 110–124.
2010. Windows Live Messenger: A Short History, Retrieved from: http://windowsteamblog.com/windowslive/b/windowslive/archive/2010/02/09/windows-live-messenger-a-short-history.aspx
Zhang, A.N., Goh, M., Meng, F., 2011. Conceptual modelling for supply chain inventory visibility. Int. J. Prod. Econ. 133 (2), 578–585.

Spyros Sioutas was born in Greece, in 1975. He graduated from the Computer Engineering and Informatics Department (CEID), School of Engineering, University of Patras, in December 1997. He received his Ph.D. degree from the same Department in 2002. Now, he is Associate Professor in Computer Science Department of Ionian University. His current research interests include: Algorithmic Data Management, Spatio-Temporal Database Systems, Distributed Data Structures and P2P Overlays, Cloud Infrastructures, Indexing, Query Processing and Query Optimization. He has published over 90 papers in various scientific journals and refereed conferences (amongst others SIGMOD Record, Algorithmica, Computer Journal, Data and Knowledge Engineering, Journal of Descrete Algorithms, Distributed and Parallel Databases, Journal of Systems and Software, ESA, ICALP, ISAAC, DEXA, SAC, CIKM, PODC, ICDT/EDBT, SODA, SIGMOD etc.). He has also more than 500 citations. He was selected to attend as Postdoc Researcher the summer school on "massive data sets" in BRICS (Basic Research in Computer Science) department, University of Aarhus, Denmark, June 2002. He was visiting researcher in Kings College, University of London (research domain: String Algorithmics and I/O complexity), London, England, April 2005. He was also visiting researcher in MADALGO (research domain: "Deterministic Data Structures over P2P Networks"), Aarhus, Denmark, December 2008. He has 10 years working experience as a Developer, Software Tester, Database Administrator and Project Manager at Computer Technology Institute (Research Unit 5) and DBIS Lab (http://di.ionio.gr/dbislab).

Evangelos Sakkopoulos graduated from the Computer Engineering and Informatics Department (CEID), School of Engineering, University of Patras and received his Ph.D. degree from the same Department. Now, he is Adjunct Professor in Hellenic Open University. His current research interests include: Software Engineering focused on Internet Technologies, Advanced Programming, Programming of Mobile Web Applications, Information Retrieval and Web Searching, P2P Data Management, Data Structures, Information Systems, Educational Technologies, Personalization and Adaptive Systems. He has published over 70 papers in various scientific journals and refereed conferences. He has 10 years working experience as a Developer, Software Tester, Database Administrator and Project Manager at Computer Technology Institute (Research Unit 5) and more than 400 citations.

Alexandros Panaretos was born in Corfu, Greece in 1979. He is currently a Ph.D. Student at the Department of Informatics, Ionian University. He obtained his B.Eng. in Software Engineering from the Computer Science Department, University of Wales Aberystwyth in 2001 and his M.Sc. in E-Commerce Technology from the Computer Science Department, University of Essex in 2002. His research interests focuses on Distributed Data Structures, P2P Data Management, Spatio-Temporal Database Systems, and Social Networks.

Dr. Dimitrios Tsoumakos is an Assistant Professor in the Department of Informatics of the Ionian University. He is also a senior researcher at the Computing Systems Laboratory of the National Technical University of Athens (NTUA). He received his Diploma in Electrical and Computer Engineering from NTUA in 1999, joined the graduate program in Computer Sciences at the University of Maryland in 2000, where he received his M.Sc. (2002) and Ph.D. (2006).

His research interests relate to Distributed, Large-Scale Systems. Currently, he is working on big-data management as well as systems aspects of Cloud Computing: RDF graphs, indexing and query processing for NoSQL, cloud application elasticity, profiling Cloud applications and platforms, etc. A full list of his publications can be found at http://www.cslab.ntua.gr/~dtsouma/pubs.html.

Panagiotis Gerolymatos was born in Greece, in 1980. He graduated from the Department of Mathematics, University of Patras in December 2003. He received his Master of Science in Digital Systems and Wireless Networks from the University of Piraeus in 2008 and his Master of Science in Information Systems from the Ionian University in 2013. He is currently a Ph.D. student in the Computer Science Department at the Ionian University having previously worked at Computer Technology Institute and Press "Diophantus", on several projects as a system engineer and senior web developer. His current research interests include: Spatio-Temporal Database Systems, Distributed Data Structures and P2P Overlays, Cloud Infrastructures, Indexing, Computational Geometry, Intelligent Information Systems and Web Services.

Dr. Giannis Tzimas is an Assistant Professor in the Department of Computer and Informatics Engineering of the Technological Educational Institute of Western Greece. From 2009 till now he is also the head of the Networks Operation Center of the Institute. Since 1995, he is an adjoin researcher in the Graphics, Multimedia and GIS Lab, Department of Computer Engineering and Informatics of the University of Patras, while from 1996 till 2011 he was also the technical coordinator of the Internet Technologies and Multimedia Research Unit of the Computer Technology Institute and Press "Diophantus". He has participated in the management and development of a large number of Research and Development projects funded by national and EU resources, as well as the private sector. His research activity lies in the areas of Computer Networks and Applications, Web Engineering, Web Modeling, Mobile Computing, Semantic Web and Bioinformatics.

Yannis Manolopoulos is Professor with the Department of Informatics of the Aristotle University of Thessaloniki. He has been with the University of Toronto, the University of Maryland at College Park and the University of Cyprus. He has also served as Rector of the University of Western Macedonia in Greece, Head of his own department, and Vice-Chair of the Greek Computer Society. His research interest focuses in Data Management. He has co-authored 5 monographs and 8 textbooks in Greek, as well as ~300 journal and conference papers. He has received >8500 citations from >1200 distinct academic institutions (h-index = 42). He has also received 3 best paper awards from SIGMOD, ECML/PKDD and MEDES conferences and has been invited as keynote speaker in 10 international events. He has served as main co-organizer of several major conferences (among others): ADBIS 2002, SSTD 2003, SSDBM 2004, ICEIS 2006, EANN 2007, ICANN 2010, AIAI 2012, WISE 2013, CAISE 2014, MEDI 2015. He has also acted as evaluator for funding agencies in Austria, Canada, Cyprus, Czech Republic, Estonia, EU, Hong-Kong, Georgia, Greece, Israel, Italy and Russia. Currently, he serves in the Editorial Boards of (among others) The VLDB Journal, The World Wide Web Journal, The Computer Journal.