Deliverable D2.4.3

Testbed-based Experimental Evaluation of Application Prototypes
Start date of the project: January 2004
Duration: 48 months
Project Coordinator: Prof. Dr. math. Friedhelm Meyer auf der Heide
Heinz Nixdorf Institute, University of Paderborn, Germany

Due date of deliverable: December 2005
Actual submission date: January 2006
Dissemination level: PU – public

Work Package 2.4: Integrated Application Testbed (IAT)
Participants: Telecom Italia Learning Services (TILS), Italy
Authors of deliverable: Federico Morabito (federico.morabito@telecomitalia.it)
Fabrizio Davide (fabrizio.davide@telecomitalia.it)
Giovanni Cortese (g.cortese@computer.org)
1 Introduction

The work performed under 2.4 aims at the implementation and evaluation of applications prototypes, while at the same time researching evaluation methods and techniques, which are suitable for understanding the behaviour of large scale, overlay based applications. Evaluation methods should allow to compare innovative designs for large scale, self-organising IT infrastructures, and analyse their properties, using both traditional measurement approaches and non-conventional evaluation metrics based on conceptual tools borrowed from complex systems research. Overall, evaluation of large-scale distributed prototypes remains a challenging issue. Often, analytical and simulation based studies of algorithms and designs provide misleading results, which are sometimes disproved when the design is prototyped and deployed in a testbed. Unfortunately, in order to perform testbed-based evaluation there are lots of practical problems (engineering, logistics etc) to be faced, as well as methodological and scientific problems related to analysis of peculiar characteristics of this type of systems. Our work confirmed these challenges. Our efforts aim at developing a better understanding of the approaches which should be used to validate this class of systems, their feasibility and cost.

We focused our work on the evaluation of a particular class of peer-to-peer systems, having the characteristics of the network management platform described in Deliverable 2.3.2. In Deliverable 2.3.2, we argue that one of the primary requirements, perhaps the most critical, for a network management platform is the ability to realise a highly scalable, decentralised, self-organising data management overlay. Typical applications related to network management, which require such decentralised data management infrastructure, include e.g. information directories, distributed data collection and monitoring, content replication and distribution. The current implementation of the management platform, which includes an integrated software implementation for a subset of the mechanisms researched in WP2.3, is reported in this deliverable.

In Deliverable 2.3.2 we report our research on large-scale distributed data management driven by the needs of network monitoring applications. We have also outlined in Deliverable 2.3.2 two main architectures being investigated for the purpose. Structured overlays have emerged mainly in an attempt to address the scalability issues that the first generation unstructured systems were originally faced with. These systems essentially provide a mapping between content and location, in the form of distributed routing indices, so that queries can be efficiently routed to the node hosting the desired content. They offer a scalable and self-organising substrate for data dissemination and search-oriented applications such as the envisioned management platform. Their main reported disadvantage for the same kind of applications is their apparent unsuitability for creating and maintaining an overlay structure which is properly load balanced, in case of uneven data distributions. Structured overlays require additional functionality (such as an extra layer supporting storage and indexing of data) to properly behave as data overlays, that is as a substrate suitable as support for implementing sophisticated search-oriented applications. In unstructured peer-to-peer networks, the placement of content is completely unrelated to the overlay topology. The content needs to be located and the searching mechanisms range from flooding the network with propagating queries to more sophisticated random walks and routing indices. This architecture is more appropriate for managing highly-transient node populations, while the search mechanisms have implications on the availability and scalability properties of the system.

In current state-of-the-art, realisation of layered applications (such as information directories or network data collection) using the unstructured approach would require in our view still requires further development in the basic mechanisms. For this reason the implementation and evaluation activities in this WP are more oriented to the issues of structured overlays (which was selected as the architecture for the network management platform).

Specific objectives of WP 2.4, as reflected by activities reported in this deliverable, include:

- To perform practical, testbed-based evaluation activities on application prototypes. Specifically,
we have performed evaluation of the prototypes both in LAN-based clusters of PCs and on PlanetLab.

- To improve our understanding of the methods for evaluating realistic P2P, overlay-based applications and devise guidelines. To this end, we have performed and included here a survey of existing approaches on evaluating dynamic properties of a P2P system.

The deliverable is structured as follows. In section 2 we describe the current implementation of the management platform. The platform prototype both defines the overall system model of the system, for which we aim at defining methodological evaluation guidelines, and is the actual software we deployed in the testbed.

2 Prototype description

One of the objectives of SP2 is to design an integrated prototype ("management platform") implementing a subset of the theoretical results of the workpackage 2.3. The main objective in year 2 is to design a peer-to-peer data management architecture, supporting ontological reasoning and peer-to-peer data management features. We provide here a description of the cross-activities WP2.3 - WP2.4 leading to the XDM platform.

2.1 Overview

The current prototype is a Java-based platform for decentralized, cooperative data management realized by a network of peers, with a specific orientation to the realization of applications such as information directories and network monitoring. It is built on top of a DHT self-organising communication overlay. The platform results from the composition of the functionalities of XDM (TILS), a peer-to-peer service supporting XML data storage and query, and the RDF Schema storage and query component (UPB) aimed at metadata management. Together, the components aim towards realization of an 'emergent' data management system. In such system, we have a set of peers cooperate to realize a distributed repository of persistent data, where each peer serves both as a data source and as a place to store data. Peers can either insert their data directly in the common storage, or they can insert index information, to guide queries to find data stored locally by the peer. This is especially interesting for data which is generated on demand (e.g. via a Web Service), or which changes frequently (e.g. a performance counter in a network monitoring agent). The peers are not required to share a common ontology or schema. In contrast, each peer may have its own way to describe data and information. In order to be able to interoperate, it is necessary that peers describe their data (i.e. they provide metadata or schema information). This may either be done automatically via usage of Data Guides or similar mechanisms, or the peer schema might have been inserted a priori and manually.

We provide here an overview of the implemented data management platform functionalities. While the following description includes details which may apparently go beyond the scope of a Complex System initiative, we believe the platform is an important achievement which will be leveraged further in the project. The platform is built on the principles of self-organisation, and implements with acceptable engineering quality the main required building blocks of a peer-to-peer data management platform. It can be used in the remainder of the project as a basis for large scale experiments, allowing execution of trials in a testbed of realistic application scenarios in the network management domain. Also, it will be used as a baseline on which new advanced functionalities and more experimental designs for self-managing distributed data management can be plugged in.
2.2 XDM

XDM implements a distributed repository for XML data. The data is stored physically in the nodes of a structured p2p network based on DHT (FreePastry is used for the DHT layer). XDM provides to client applications two main APIs for disseminating and querying information a) a query interface over persistently stored, replicated data b) a publish-subscribe interface over non persistent data. Update over persistent data is supported. Both are based on an XML view of the data and XPath-style of expressing queries or subscription patterns.

2.2.1 Distribution Model

Applications can store data (XML fragments) in the network of peers. The unit of distribution is an XML fragment associated with a 'storage type' (we may use type for brevity in the following text). A type definition is a collection of metadata which provide directives to the distributed storage manager on how to store the XML fragment. Such directives include criteria for forming the primary key ('id') of the XML fragment starting from XML attributes and elements; for indexing the fragment based on additional attributes or elements; and for selecting the granularity of data storage. With respect to granularity, to achieve desirable scalability, storage types can be one of 'single' (where each fragment, as identified by its id, maps to a single peer and storage object within the peer), 'log' (where items identified by the same id are stored at the same peer, and appended to a 'queue', as appropriate for example for a time series of observations reported by a sensor or network management agent), 'vector' (where multiple data items with same primary id, are stored at the same peer, while being individually addressable for selections or updates based on a vector key).

2.2.2 Replication

Also, the storage type definition provide hints on the desired replication level which is applied to the XML fragments. A leafset-based replication manager implemented as part of XDM provides for replicating the data, and for maintaining the desired level of replication, in spite of events such as departure or failure of nodes.

2.2.3 Indexes

XDM includes an indexing functionality, which is layered atop a DHT. Index objects, together with the value of the indexed attribute, contain a URL to the indexed fragment. XDM supports three index types: equality-predicate indexes, range-predicate indexes, and type-predicate indexes. Equality-predicate indexes on primary key are mapped to DHT lookup operations. Range-predicate indexes aim at supporting range queries; the adopted implementation scheme currently rely on a priori domain partitioning into intervals, which can be based on uniform or other expected distribution of values for the attribute to be indexed. Type-predicate make it possible to process queries which do not include any condition on indexed attributes without broadcasting the query to all nodes. It is well known that 'popular' values (or ranges in the case of range queries) for indexed attributes may yield load unbalance in the system, by causing large index objects to be stored on the same peer. To address scalability, index objects are implemented as B+Trees, and store up to a maximum number of entries. While this design still suffers from having the root of the B-tree involved in processing most queries, it provides a simple and robust solution to load balancing. Index objects are also replicated and garbage collected using both a reference count mechanism, and a periodic background scan to recover space. Ongoing work is targeting tries [Prefix Hash Tree] to provide generalized implementation of range indexes.
2.2.4 Index vs plain data in the DHT

In XDM, there are two possible usage scenarios (and a combination thereof) for the distributed storage infrastructure. With the first, the emphasis is on using the distributed storage infrastructure to store data. Data is replicated to ensure availability even when node which originated it is not available. In the second usage scenario, data is not stored in the DHT but locally stored on the peer which is the source for the data (this is the case, for example, of logs of data generated by sensor nodes or network management agents). Data is not replicated; on the other hand, the DHT is used to index all available data in the system. A 'local storage' directive, which can be selected for a type, provides the programmer with convenient API for selecting this usage scenario.

2.2.5 Query processing

Retrieval of data is performed by specifying a storage, a type and an Xpath expression specifying the matching documents for this datatype. The expression can be the primary key of the object which is to be retrieved (this may yield a list of documents if the datatype is of type "log"; in this case an entryid can be specified to retrieve a specific version). Another method of querying is to give an XPath expression which has to be matched by the documents to be returned. There are two ways for performing retrieval, one exploits primary key (which implies a DHT lookup), the second exploit indexes. The query processor is currently simple, does not perform join-type operations in the network and uses simple strategies for index selection. A subset of Xpath is currently supported for expressing queries over the distributed repository.

2.2.6 Storage Catalog

All metadata (type definitions, index definitions, topics et al) in XDM is described in XML and stored in the DHT. Similarly, schema information in the form of automatically generated dataguides [DG2], [DG1] or XML Schemas is being included in the storage catalog (integration work ongoing).

2.2.7 Publish-Subscribe

XDM includes a topic-based publish-subscribe system which leverages prefix routing of Plaxton trees (implemented as an extension of FreePastry Scribe). Beyond being an application level service, publish subscribe is used internally for distributing management and metadata information. A planned activity is an implementation of the solution described into [DELIS-TR-0228] to provide an implementation of adaptive, content-based publish subscribe.

2.3 Metada management

The UPB component provides for metadata management. By supporting RDF Schema and taxonomical reasoning, it allows to represent network management ontologies in the peer-to-peer network. By using the facilities of this component, clients can figure out and reason about the schema of data stored in the network. We briefly describe the architecture of this prototype.

See Figure 1 for an overview. Each node takes care of several local information sources. The node collects the RDF triples in a local storage. No reasoning is done with this knowledge. The triples are directly sent to the relevant nodes in the network, according to the distribution described in section 1.3. The triples which are received via the p2p network are stored in a separate container. The evaluation of the RDFS rules is done with respect to the triples stored in this container, which again leads to the generation and distribution of new triples. The query evaluation algorithm uses the knowledge stored in the network by sending out messages to the appropriate nodes. The local knowledge is accessed transparently in the same way. The prototype is based on the FreePastry implementation of the Pastry DHT network [Pastry].
2.4 Integration of components

We are currently working at integration of the components. The first integration step targets the implementation of an integrated query client which uses schema information available in the network to help users in creating his own query. Further integration work planned will target mechanisms for query expansion, and research on mediation techniques for query processing.

3 Evaluating emergent behaviour and autonomic properties of overlay-based applications

3.1 Introduction

As outlined in section 1, we are researching methods to improve our ability to predict the behavior of large, dynamic systems before deploying them to a large set of users. This requires in the long term a combination of analytical, simulation-based evaluation and testbed-based evaluation. An important concern here is developing methods and metrics for evaluating emergent properties of decentralized systems. The very nature of large-scale systems with concurrent and interacting entities, make it intrinsically hard to predict the overall behaviour of the system. The dynamicity of the environment amplifies the situation. From a scientific viewpoint, the challenge is to avoid traditional trial-and-error approaches, and propose well-founded and mathematical tools to investigate collective emergent behaviour. From the viewpoint of industrial acceptability of decentralized overlay-based applications, it is critical that application developers be empowered with methods and tools for assessing properties such as robustness, resilience and stability of the system in simulated or controlled testbed scenarios, before the application is deployed to the users. Highly decentralised, peer-to-peer architectures will only become industrially acceptable if the developers can provide well-founded statements on the overall behaviour under particular forms of dynamics. Examples of dynamics include heavy load variations, departure or disconnection of nodes from the overlay, network partitioning of the overlay, and how the addition and removal of components impact application behaviour. Evaluation methods should allow assessing system properties such as, for example:

- QoS: although heavily application dependent, QoS metrics allows to measure how the system provides the intended service to users
• Robustness / failure avoidance: Ability to provide the service in spite of different types of failure

• Stabilization Time: time needed to return to some ‘stable’ state, and provide acceptable quality of service, after some failure

• Sensitivity: the degree of impact of changes at the micro-level on the behaviour at the macro-level

Some of the topics above require a theoretical perspective drawing e.g. on complex networks theory, statistical mechanics or chaos theory. There are several types of system-wide properties which may require different means or techniques to be evaluated. A widely accepted taxonomy of system-wide properties relevant to evaluation of overlay-based applications would be a beneficial first step to establish consistent evaluation practices; our survey however did not reveal much ongoing work in this area. We informally summarize in 3.1.1 some of the relevant concerns:

1. Efficiency of queries and notification of data to subscribers.

2. Robustness of the system and its ability to conduct in a ”stable state” the system despite perturbations.

3. Load balancing, that is the ability of the system to balance the overall load of the nodes under acceptable level of efficiency.

4. Scalability with respect to nodes and data.

In the remainder of section 3, we provide a survey of approaches to evaluation of large-scale overlays, with emphasis on existing works that leverage techniques from complex systems research.

3.1.1 Efficiency

Efficiency is in the first place captured by QoS properties as perceived by clients, that is latency of important client operations. Such operations for a middleware such as XDM include (respectively, the first 2 are for persistent storage operations while the latter are for publish-subscribe communication):

1. Insert operations (storing an XML document).

2. Query response time.

3. Notification of a published data to a subscriber.

4. Processing time of a subscription.

Average and variance of the selected variables will be of interest. Notice that for asynchronous operations, such as notification, collecting accurate measurements can be less than easy. Beside latency, we are interested in analysing overall system throughput. Metrics in this area include the number of events that can be delivered per unit of time using multicast, publish-subscribe communication; also the number of data that can be inserted in the system, as well as the number of concurrent queries (Note: this could be seen as a scalability issue). Finally, we are interested to study the overall efficiency of the protocols underlying the client services, and collect information such as the overall bandwidth consumed by the services (broken down by the different operations of the protocol, including background operations such as replica maintenance, index maintenance and so forth).
3.1.2 Robustness

Peer-to-peer systems (such as the data overlays we are concerned with) aim at providing guarantees out of an unsecured, untrusted set of computing and communication resources. Large-scale correlated failures can happen in such network of peers. A data management network needs to maintain availability of the stored data despite large-scale correlated failures. The system must be then designed to prevent data loss even under extreme conditions, such as correlated failures with data loss on high percentages of the storage nodes, at the expense of extra storage cost. The availability (usually, the fraction of time under which data is available) and durability (the ability to survive crashes) guarantees for data, that must be delivered, are specific to applications, and can be very high in some cases [Bhan]. Robustness is usually achieved through mechanisms of data replication and migration. Data replication is a technique that duplicates in an optimized way the content within the nodes of the system to increase the availability of the data. In adaptive systems, data replication mechanisms spawn multiple copies of the data in accordance with the dynamics of the requests: high demand increases the replication factors, while a low demand decreases these factors. By increasing the number of replica without coordination among the nodes, the system could lead to high level of availability of data, but on the contrary, the system performance could be deteriorated when a significant number of nodes leave the system. In such condition, large number of copies start to migrate to different "alive nodes", with the effects to saturate the available resources (both bandwidth and storage). The effect of data replication without a limit on the number of replicas (or without a resources coordination mechanism) could be, of course, negative effects also on the efficiency on propagation and distribution of the information, that is the second features. Data migration techniques push the content in those nodes that better can accommodate the data and that are closer to the appropriate location. For example, data migration techniques are useful when the local demand of the content is known or can be estimated. For example, a content provider can known a priori the request of some data or in a network management scenario, the local demand can be forecasted by the class of subscriptions that the monitor nodes have been already submitted. Under a fixed and limited number of replicas that is lower than the actual demand, the local demand will be served under non-optimal solution. Also, free-riding behaviours of some malicious node that issue the most request for a data without the needed of such number of data could lead to an un-balanced distribution of the data closer to such node.

Both techniques affect the capacity of the system and stress the resources with the degradation of the performance of the system. To design an efficient data management infrastructure that makes available the data and store the appropriate number of replica and, at the same time, does not saturate the resources, making possible the effective distribution of information, passes through the trade-off between data replication and data migration approaches. At the same time, the design of infrastructures that will be based on such approaches require to introduce advanced monitoring tools in order to characterize also the dynamics of the data. In this work, we will study methods and techniques that can be applied for monitoring and evaluating the data movement.

3.1.3 Load Balancing

Another important property that data management overlays strive to achieve is the ability to balance the load (storage and query processing load) among the nodes. The load on individual nodes should be balanced to ensure overall fairness and should be increased or decreased based on the current workload. The measurement of the active transactions in which each node is involved allows estimation of the local load. As for other properties, mechanisms for collecting global load are necessary (e.g. a data aggregation service). As results of a combination of factors the distribution of data and query loads might become skewed, and some sites might become overloaded whilst others might be idle (load imbalance). The imbalance ratio is defined as the ratio of the loads of the largest
and smallest loaded site. The load imbalance ratio is a metric which can be used to represent load balancing for both storage and query processing/data dissemination load, and should be studied in test scenarios including stable system and systems with dynamics.

3.1.4 Scalability

Scalability is obviously a central requirement of the design of overlay-based applications. Scalability tests should verify that the system performance degrades ‘slowly’ with an increase in the workload and stored data in the overlay network.

3.2 Evaluating overlay networks construction and maintenance

A common framework for evaluating the performance and self-* properties of overlay-based applications is still lacking. We report briefly some common issues that have to be considered during the evaluation of a system based on overlay networks [Dabek]. A list of factors that affect the performance at the overlay network layer are the following:

- **Churn** - is the most related to our research perspective; see discussion below.
- **Packet loss** - host availability is strongly dependent on the time-varying and transient packet loss function.
- **Proximity routing** - with proximity routing, the routing choice takes into account the network latency to reach a neighbouring node. Various algorithms implement proximity routing differently, but they all adopt the same basic approach of weighing progress in identifier space against cost in latency (or geography). Such way of routing should be take in account also in a internet-like environment (and not just during simulation) because the performance of the routing could be degraded.
- **Caching** - The caching techniques deeply affect the performance on search operations.
- **Latency**.
- **Number of hops** - Latency and number of hops have a strong impact on the response time of the application.

Churn (a.k.a. system dynamics) arises from continued and rapid arrival and failure (or departure) of a large number of participants in the system, and traces from deployments have shown that it can lead to extremely stressful networking conditions. It has the potential to increase host loads and block a large fraction of normal insert and lookup operations in the peer-to-peer system [Li]. Churn has obviously a major impact on most performance indicators of a P2P system, and its careful study is not an easy task. Most studies of the effects of churn we know, focus their attention on the impact of churn on connectivity/routing table optimality, and the overhead required to maintain and keep the routing overlay up-to-date. We suggest another major issue, which is of paramount importance for data overlays and for the type of applications we are studying, to be the cost inherent in moving data around, to keep up with node join and leave (churn). In fact, applications such as p2p data repositories may require each peer to store large amounts of data. This problem is amplified by the need to make extensive use of data replication, for the data management overlay to provide appropriate availability and performance guarantees.
3.3 Coping with scalability: a physics-based approach

[El-Ansary] describes initial work in applying a physics-style approach for analyzing the behavior of structured overlay networks that implement self-organization and self-repair policies. Based on the observation that simulation can be prohibitive for such systems, this work aims at making simulation more scalable by characterizing a system using intensive variables, i.e. variables that are independent of the size of the system. (Note: Self-organization in a data overlay refers to the ability of creating a communication overlay graph with the desired diameter and in/out-degree; similarly, to balance storage distribution among all nodes. Self-repair addresses what we call dynamic behavior of the system, i.e. its ability to keep the desired properties under node join/leave (‘churn’).) In physics, extensive variables are those that eventually become proportional to the size of the system, such as total energy. Intensive variables, such as density, temperature and pressure, on the other hand, are independent of system size. Authors of this work identify two cases for intensive variables when modeling a structured overlay network. First, they identify density of nodes in a SON as the ratio between the number of nodes and the identifier space; based on that, they study characteristic behavior of the average path length depending of density. By running simulation, they show that for any given number of nodes, the average lookup length increases when they are placed in a smaller identifier space and the relative effect is the same irrespective of the system size, thus proving density to be an intensive variable. Then, they study the ratio of perturbation (indicator based on frequency of join and leave events in the system) to stabilization (the time needed to a node to repair its edges of the overlay graph). They show that an equilibrium between these two competing forces holds, and scale with the system size. This result allows to conclude that the distance from optimality (number of wrong pointers) in the overlay depends on an intensive variable related to a function of perturbation and stabilization, thus making simpler the construction of evaluation models for any performance metrics that is influenced by system dynamics. Further developments along this physics-oriented way of thinking may include the study of phases of the system.

3.4 Using structural properties of the overlay communication graph

A different, and interesting approach for evaluating self-* properties of overlay-based applications is that used in [Jela] for the purpose of analysing protocols for (random) peer sampling. In this approach, authors analyze some properties of the protocol (such as randomness and uniformity of peer selection) using a graph theoretical framework, by studying the structural properties of the overlay graph that results from the gossip/view exchange protocol. Authors claim that using a graph theoretical approach provides richer possibilities for interpretation in terms of understanding the robustness, reliability of the protocol etc. with respect to a statistical approach. Example structural properties that are used in the analysis include degree distribution of the overlay communication graph, including average degree, the dynamics of the degree of a node, and the exact degree distribution.

Using this approach, they examine if these overlays exhibit stable properties; they also measure the extent to which these communication topologies deviate from the desirable uniform random model. They do so by looking at several static and dynamic properties: degree distribution, average path length and clustering coefficient. Finally, they consider the reliability of the service by examining its self-healing capacity and robustness to failure. This is accomplished by studying the connectivity properties of the graph that results from simulating massive node failures, as well as the time required to rebuild a properly connected overlay. While the scope of this work is clearly different from ours, we believe analysis of structural properties of overlay graph may provide interesting insights into the self-* properties of distributed applications; we will consider applicability of these ideas to peer-to-peer storage systems and publish-subscribe systems in the remainder of our work. Devising methods for collecting efficient properties of the network in a running system (i.e. in the context of testing in
An important part of the activities in this WP are represented by practical testbed-based evaluation activities. Testbed-based evaluation provides crucial insight such as the impact of engineering concerns over more theoretical and algorithmic aspects, and may reveal the importance of concerns whose impact has been left out in an idealised simulation model. The testbed should reproduce the internet-like working conditions. Testbed evaluation activities that we performed include:

- Set-up of a stable configuration of the prototype developed (see section 2) in a testbed based on a cluster of LAN-connected PCs. The setup activity included study and installation of Modelnet (http://issg.cs.duke.edu/modelnet.html). Modelnet allows to build an emulation environment for wide-area systems. Since in cluster-based deployment the network is usually highly over-provisioned and unrealistically uniform in its performance characteristics with respect to real Internet environment, Modelnet (a software running on one or more machines) allows to route packets flowing between the peers of the overlay as if the peers were connected through the Internet. It allows to emulate several network topologies, thus tracking the effects of congestion and competition among competing packets.

- Setup of a stable configuration of the prototype developed (see section 2) in the PlanetLab network. PlanetLab is a geographically distributed testbed designed to support the deployment and evaluation of planetary-scale network services.

- Instrumentation of the XDM components with code to collect interesting 'local' variables.

- Definition of a test plan, and development of client software to emulate several query loads foreseen within the plan. The plan includes several test suites, including suites for a) functional test, b) performance and scalability tests in static scenario, and c) performance tests in dynamic scenario. Metrics to be collected during the tests are an initial set derived from the rationale in section 3.1.

- Execution of tests and analysis of results.

At this stage, our initial experience has been in evaluating the functional aspects of the prototype. After some regression test and re-engineering of some functionality, the prototype is now stable on all functionalities. We plan to continue working on the described set-up and progressively execute more complex test scenarios, and refine the collection and analysis methodologies accordingly.

5 Conclusions and Further Work

The report describes our work in progress towards development of well-understood practices for testing peer-to-peer overlay applications. The agenda for further work is rich. Work planned in the forthcoming period include:

- Further execution of tests.

- Design and implementation of an aggregation service as a building block of the overlay functionality. The aggregation service will support aggregation queries for system-wide performance metrics (such as average or extremal values for load) thus making easier data collection for system evaluation.
• Research into a taxonomy of metrics, and methodology for practical collection and computation of such metrics in distributed overlay-based services.

• Research on methods for estimation of global, system-wide performance metrics (Distributed Statistics).

• Integration of new functionalities and improvement of current algorithms in the platform; integration of TILS and UPB components

References


[El-Ansary] Sameh El-Ansary, Erik Aurell, Per Brand and Seif Haridi Experience with a physics-style approach for the study of self properties in structured overlay networks.


[Dabek] Frank Dabek, Jinyang Li, Emil Sit, James Robertson, M. Frans Kaashoek, Robert Morris Designing a DHT for low latency and high throughput

[Li] Jinyang Li, Jeremy Stribling, Thomer M. Gil, Robert Morris, M. Frans Kaashoek Comparing the performance of distributed hash tables under churn

[Bhan] Ranjita Bhagwan, Stefan Savage and Geoffrey M. Voelker. Understanding Availability

[Yala] Praveen Yalagandula and Mike Dahlin. A Scalable Distributed Information Management System


[DG2] Felix Weigel Holger Content-Aware DataGuides for Indexing Large Collections of XML Documents

[DG1] Roy Goldman and Jennifer Widom DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases

[Prefix Hash Tree] Sriram Ramabhadran, Joseph Hellerstein, Sylvia Ratnasamy, Scott Shenker An Indexing Data Structure over Distributed Hash Tables