Deliverable D6.3.2

Final Report on
P2P Infrastructures for Efficient Information Dissemination
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 6.3: Strategies for Self-Organizing Information Dissemination and Aggregation

Participants: Research Academic Computer Technology Institute (CTI), Patras, Greece
Department of Computer Science, Universita di Bologna (UniBO), Italy
Wroclaw University of Technology (TU Wroclaw), Poland
Telenor Communication AS (Telenor), Oslo, Norway

Authors of deliverable: Peter Triantafillou (peter@ceid.upatras.gr)
Nikos Ntarmos (ntarmos@ceid.upatras.gr)
Ioannis Aekaterinidis (aikater@ceid.upatras.gr)
Theoni Pitoura (pitoura@ceid.upatras.gr)
1 Introduction and Overview

Work Package D6.3 is primarily concerned with strategies for self-organizing information dissemination and aggregation. The work conducted within DELIS for D6.3.2 constitutes the final report for our work on P2P infrastructures for information dissemination and aggregation. This report combines the intermediate reports submitted a year ago for work packages WP6.3 and WP6.5, which were integrated in the new DELIS implementation plan into a new work package, WP6.3, and extends our previous report with our new results.

Our efforts have focused on addressing open problems for information dissemination and aggregation within structured peer-to-peer (p2p) data networks and especially with networks based on Distributed Hash Tables (DHTs) [61, 51, 20, 30]. Our main contributions are centered on the following two pillars:

- developing architectures and algorithms for processing complex information processing requests over DHTs in an efficient and scalable manner and on
- developing architectures, statistical structures, and associated algorithms using which one can scalably and efficiently aggregate and extract valuable information defining the global state of the data network.

In particular, with respect to our work on processing complex information processing queries over DHTs, we

- developed architectures and algorithms to support publish/subscribe information dissemination networks, based on DHTs, [4] [63]. Publish/subscribe systems involve both numerical and string attributes and with our contributions we show how to support rich queries on string attributes (supporting prefix, suffix, and containment operations) over DHT-based networks.
- developed solutions for efficiently processing range queries over DHT-based networks, [41]. Range queries have been neglected by the original DHT network architectures that only cater to simple, exact-match lookups and this has spurred considerable research into supporting range queries.
- developed a new architecture that can address on the one hand the load balancing problem using the replication of hot data and, on the other, provide an efficient support for range queries [49]. This combined problem is particularly challenging. The replication of data yields load balancing since it allows different replicas to be randomly accessed. However, the efficient support of range queries is accomplished using clever data placement of consecutive values on the network. Thus, the random accesses imposed by data replication to ensure load balancing interfere with any clever data placement techniques used to support efficiently range queries. Our efforts have resulted into HotRoD: an architecture and related algorithms for replication management and utilization for load balancing and algorithms for efficient processing of range queries.

With respect to our work on aggregating and extracting key (meta)information for the data network, we have made the following contributions.

- In joint work with MPII Saarbruecken, we developed a highly distributed, scalable counter, primarily for estimating the cardinality of (multi-)sets in both a duplicate sensitive and duplicate insensitive manner. This tool, coined Distributed Hash Sketches (DHS), [44] can be utilized in a variety of key applications, providing highly accurate estimations for a large number of metrics, ranging from estimating the size of the network, the total number of data objects with a particular interesting property, etc.
We additionally focused on (i) proposing novel statistical metrics for the measurement/evaluation of load distribution in the nodes of a DHT and (ii) developed associated sampling algorithms for their online, efficient, and scalable computation [48]. Together these contributions can yield important new insights as to the characteristics of the load distribution at any point in time, and help to guide load balancing decisions, such as whether load balancing activities should be invoked and for how long.

We presented SeAl [43]: a monitoring and auditing layer, which can be used to detect key, network-wide information, such as the altruistic/selfish behavior of nodes of a DHT. SeAl is capable of being deployed in both structured and unstructured P2P networks, featuring multiple knobs so to be tailored to the system’s needs.

In [42] we presented AESOP a routing infrastructure building upon currently available state-of-the-art routing technologies (such as Distributed Hash Tables) and a version of SeAl (coined AltSeAl) specially tweaked to incur minimum maintenance overhead.

Both of these threads of work are expected to be of real use for other work packages in SP6 of DELIS and, in particular, for helping to meet central challenges toward the development of a large-scale, highly-distributed search engine, which is the champion application to be developed by SP6.

2 Main Results on (Meta)Information Aggregation in Internet-Scale Data Networks

2.1 Estimating MultiSet Cardinalities in DHTs

2.1.1 The Problem

Frequently, counting in general, and estimating the cardinality of (multi-) sets in particular, is highly desirable for a large variety of applications, representing a foundational block for the efficient deployment and access of emerging internet-scale information systems. Examples of such applications range from optimizing query access plans in internet-scale data networks, to evaluating the significance (rank/score) of various data items in distributed information retrieval applications. The key constraints that any acceptable solution must satisfy are: (i) **efficiency**: the number of nodes that need be contacted for counting purposes must be small in order to enjoy small latency and bandwidth requirements; (ii) **scalability**, seemingly contradicting the efficiency goal: arbitrarily large numbers of nodes may need to add elements to a (multi-) set, which dictates the need for a highly distributed solution, avoiding server-based scalability, bottleneck, and availability problems; (iii) **access and storage load balancing**: counting and related overhead chores should be distributed fairly to the nodes of the network; (iv) **accuracy**: tunable, robust (in the presence of dynamics and failures) and highly accurate estimation of the cardinality; (v) **simplicity and ease of integration**: special, solution-specific indexing structures should be avoided. In joint work between CTI and MPII in this direction, we proposed **Distributed Hash Sketch (DHS)** [44]: a novel, scalable, fully distributed approach to estimating global metrics, such as size of parameters of interest, based on hash sketches.

2.1.2 State of the Art Overview

Hash sketches constitute a probabilistic cardinality estimator for large multisets, capable for duplicate-sensitive and duplicate-insensitive counting. They were originally proposed by Flajolet and Martin[22] and were more recently extended by Durand and Flajolet[18]. Briefly, in their essence, hash sketches work as follows. A bitmap vector $R$ is used to implement counting. Inserting a data item of the multiset results in one bit of $R$ being set. The counting mechanism ensures that bit 0 of $R$ is set by
half of the multiset population, bit 1 will be set by a quarter of the population, and, in general, the $i$th bit of $R$ will be set by $\frac{1}{2^i}$ of the population. After insertion of all items, estimating the cardinality of the multiset entails finding the position $d$ of the most significant non-zero bit of $R$. Specifically, if $S$ is the cardinality, it holds that $d \approx \log_2(N)$.

Hash sketches have been extensively used recently in demanding applications, such as data streaming applications. To our knowledge, our work is the first work to consider the use of hash sketches in the context of Distributed Hash Tables.

Aggregation operations, in general, within the peer-to-peer landscape has received recently some attention. One of the first works was Astrolabe[54]; the authors proposed the creation and maintenance of a hierarchical, tree-like overlay, used to propagate complex queries and their results through the peer-to-peer overlay. Aggregation over DHTs was discussed in [31]. The authors borrow ideas from parallel databases but, as they do mention, such techniques “are not necessarily appropriate in a multi-hop overlay network”. [16] have described a method for implementing $SUM$ aggregates using hash sketches.

[8] propose building a (set of) multicast overlay tree(s) to propagate queries and results back and forth, while using flood-like methods to send messages around the network. [8] also propose to use hash sketches to estimate aggregates over a peer-to-peer network. Their scheme, coined, (MultipleTrees) exhibits a high messaging overhead, in the order of $O(E+k \times N \times \log(N))$, where $E$ is the number of edges in the P2P graph, $N$ is the number of nodes, and $k$ is the number of multicast trees built.

Other researchers have proposed aggregation techniques based mainly on gossip/epidemic like protocols[32, 35]. Although the bandwidth requirements of these approaches appear to be small, when viewed over the whole node population the overall bandwidth consumption and hop-count are usually very high, of $O(N)$. Moreover, all nodes have to actively participate in a gossip-based computation, even if it is of no interest to them.

2.1.3 Our Contributions

Distributed Hash Sketches is a novel, fully decentralized mechanism, capable of providing estimates on the cardinality of multi-sets of objects in a peer-to-peer system. DHS is, to our knowledge, the first truly distributed version of hash sketches along with the accompanying algorithms, and protocols. Moreover, it is the first distributed counting mechanism satisfying all six constraints presented earlier.

In its simplest form, a DHS is based on the following:

- The node ID space of the underlying DHT is partitioned into segments of exponentially increasing sizes.
- Each segment is associated with a single bit position of $R$.
- Specifically, the largest segment (which has a size that is one half of the node ID space) is associated with bit position zero. Similarly bit position 1 is associated with the next biggest segment, etc.
- All nodes (with IDs) falling in a particular segment collectively implement the responsibility for their associated bit position. Specifically, they each maintain a local bit.
- Adding an element into the multiset requires determining first which bit position of $R$ must be set (in a manner similar to traditional hash sketches). Then, the DHT segment associated with this bit position is found. Finally, a node from this segment is chosen randomly and it is asked to set its local bit.

3
Cardinality estimation basically involves finding the most significant segment (bit) that is non-zero. To do this, for each segment, a node is chosen randomly and its local bit is tested to determine whether the corresponding $R$ bit is zero.

In [44] a number of optimizations and alternatives are given to improve and/or trade-off accuracy of estimation for efficiency of counting.

Designing and implementing hash sketches over DHTs in an efficient and scalable way, while maintaining the, implicit in the peer-to-peer world, invariant of completely decentralized operation, is a formidable task. Our contributions include: (i) balanced access load, (ii) highly efficient and scalable operation, independent of the number of data items and logarithmic in the number of nodes in the overlay network, (iii) derivation of bounds on the error added by the distributed operation and examination of its algorithmic implications, (iv) alternative ways to implement hash sketches in a distributed manner, (v) implementation and evaluation of both [22] and [18] hash sketches within our framework, and (vi) implementation and evaluation of DHS with respect to its estimation error and overhead and with respect to utilizing DHS-based histograms for query optimization.

In addition to the above, the proposed design: (i) is DHT-agnostic, in the sense that it can be deployed over any peer-to-peer overlay conforming to the DHT abstraction, (ii) provides probabilistic guarantees regarding the correctness and accuracy of the produced estimates, (iii) allows for a trade-off between accuracy and cost of maintenance, and (iv) incurs low bandwidth, storage, and processing overheads, when used for counting the cardinality of widely distributed item (multi-)sets.

2.2 Evaluating Load Distributions in DHTs: The Gini and the Bottle

2.2.1 The Problem

This work [48] addresses the issue of fair load distribution in peer-to-peer (P2P) networks from a global perspective. Significant load imbalances arise from skewed data and query distributions, from a high degree of heterogeneity in data items’ load and node capacities, as well as from continuous nodes’ arrivals and departures, and continuous data items’ insertions and deletions. Although there are plenty and significant solutions to address the issue of fair load distribution, to our knowledge, none of them addresses it from a global perspective. Our work specifically focuses on two main problems found in related work: the metrics problem and the local-optima problem.

The metrics problem. In the P2P community what is very much lacking is the definition of appropriate metrics capturing naturally the notion of fair load distribution and providing rich, compact information about this distribution. Furthermore, and perhaps more importantly, such metrics should not be simply utilized offline, i.e. to simply test the efficacy of specific algorithms after the fact (i.e. after they ran) but, in addition to drive and guide the functionality offered by a specific algorithm online, helping the algorithm to achieve fair load distribution.

The local-optima problem. It is true that without an overall appreciation of the state of the load distribution, imbalances may cause severe inefficiency and scalability problems. Given load imbalances, corrective actions are typically based solely on local information. Thus, they may look promising from the point of view of specific nodes, while in reality being far from optimal, from a global perspective.

2.2.2 State of the Art Overview

Load balancing goals in all related work ([23], [25], [27], [34], [50], [64]) focus on minimizing maximum load and avoiding overloaded nodes in P2P networks. None of these works, however, deals with measuring the fairness of load distribution among nodes and for ensuring it. This work states and addresses the issues of the local optima and the metrics problems by proposing a global perspective to load distribution and defining inexpensive and rich load distribution metrics. It is also a generalized approach that can be applied on top of any P2P network.
2.2.3 Our Contributions

Our detailed contributions [48] are: We

- proposed and studied "inexpensive, accurate, natural and rich" load distribution metrics, in the sense that they are capturing naturally the notion of the fairness of load distributions, providing an accurate and rich global picture of them.

- developed lightweight distributed sampling algorithms to compute estimates of these metrics efficiently, scalably, and with high precision, and

- showed how these metrics can be utilized online by higher-level algorithms, which can know when and how to best intervene to correct load imbalances.

We put forward a number of requirements that any load balancing metric must meet before it can be deemed as appropriate. The particular metric we advocate is the Gini coefficient, which is based on Lorenz curves [19]. Gini was shown to be by far the most predictable metric, in the sense that observing two different estimations of it (taken at different times) one can appreciate the change that has occurred in the underlying distribution. This is very important given the high dynamics of P2P networks; all other candidate metrics varied wildly in changes in the underlying load distributions. Further, Gini was proved to enjoy the highest precision, in the sense of its estimations having the least relative standard deviation among all evaluated metrics. Subsequently, we developed and studied the performance of distributed sampling algorithms, specifically tailored to compute the Gini for various load distributions. The impressive result is that with these sampling algorithms, at the cost of approximately one DHT lookup, Gini can be estimated with high accuracy, a fact that introduces all the benefits of being able to utilize this approach online, driving any load-imbalance corrective actions.

2.3 Monitoring and Accounting for Node Behavior in P2P Data Networks

Our relevant efforts have concentrated on the design, implementation and study of SeAl. SeAl is an infrastructure transparently weavable into (structured and unstructured) P2P sharing networks. SeAl components act in two ways; they provide the system with the necessary infrastructure to categorize peers and to allow them regulated access to resources, according to their contribution to the community; and they urge users to be altruistic, in order to build up a good reputation in the system.

2.3.1 State of the Art Overview

The problems of “trust”, “reputation”, and “accountability” in distributed systems have been a research hot-spot for quite some time. However, widely deployed, web-scale, data sharing systems that maintain metadata about participating nodes ([69, 21]) don’t address the problems posed by selfish and malicious peers. Payment schemes have been suggested by several researchers for enforcing fair-sharing. However, to the authors knowledge, all existing payment schemes ([56, 13, 26, 70]) make the assumption of the existence of a globally trusted (centralized) entity at their core. Cornelli et al. ([17]), present a system to select the most reputable peer to download content from, a limited functionality compared to SeAl. GNUnet ([24]) like SeAl, allows for the exploitation of “excess” resources. However, SeAl uses a much stronger fair-play enforcement scheme, while also being completely decentralized. FreeHaven is a client-server-based system where only servers perform trust computation, not applicable to a true peer-to-peer sharing network. [40] have peers keep “usage files”, describing interactions with other peers, and peers “audit” both their neighbors and random nodes at random intervals. Their problem is trade-oriented: nodes want to store their data on other
nodes. Furthermore, they use a quota-based reasoning; “under quota” peers are excluded from the network. This leaves no space for the exploitation of excessive network/processing resources.

A work very close to ours is that of Aberer et al. ([1]). [1] also use a DHT-type overlay (i.e. P-Grid) to store transaction information. Compared to their work, SeAl: (i) is less prone to Sybil attacks since SeAl uses both black- and white-lists, keeping track of selfish and collaborative/altruistic behaviour, as opposed to only black-lists (“complaints”); (ii) features queueing and scheduling algorithms (built on top of our monitoring layer) unique to our setting, providing incentives for altruism; (iii) uses random checks and different levels of verification checks for efficiency, while the verification chores can be left aside or tuned at will to achieve better response-time efficiency; (iv) allows for inconsistent malicious/selfish behavior, different forms of punishment, and for the exploitation of excess resources ([1] use binary logic as to the behaviour exhibited by peers – a peer is either completely selfish or completely altruist – and, based on this, are allowed either full or no access to resources); (v) features an aging mechanism further addressing temporal inconsistencies and variations in user behavior.

2.3.2 Our Contributions

Conceptually, SeAl consists of two distinct layers: (i) the SeAl monitoring/accounting layer (SAL), monitoring behavior and maintaining all metadata pertinent to the peers’ participation and contribution to the rest of the community, in the form of black- and white-lists recording selfish and altruistic behavior, respectively, and (ii) the SeAl auditing/verification layer (SVL), utilizing cryptographic techniques in order to provide overwatch to the operations of the accounting layer in the presence of misbehaving users. These two layers form a substrate utilized by an incentives mechanism, which essentially increases the shared pool of content and resources. The basic idea is that all transactions between peers result in the creation of tokens (called “Transaction Receipts” or TRs) that can be used much like “favors” in real life; the peers rendering favors (i.e. sharing resources) gain the right to ask peers receiving favors to somehow pay them back in the future or get “punished” otherwise. All of these operations are performed transparently to the user.

Nodes keep track of the favors they render or receive (i.e. store the corresponding TRs) in two “favor lists”: the “Favors-Done” ($F_d$) and “Favors-Owed” ($F_o$) lists. Moreover, nodes in SeAl are characterized by their “altruism score” (denoted by $n_i.A$). This is simply a function of $|F_d|$ and $|F_o|$, where $|X|$ denotes the size of the set $X$. For example, we can consider $|F_d| - |F_o|$ or $|F_d|/|F_o|$ as possible altruism score functions. If node $n_1$ shares a resource $r_1$ and node $n_2$ accesses it, the favor-lists mechanism enables $n_1$ to selectively redirect a subsequent incoming request for $r_1$ to $n_2$. SeAl nodes autonomously and independently set an upper ($n_i.A_{max}$) and a lower ($n_i.A_{min}$) threshold value for their score. When they rate higher than $n_i.A_{max}$ they always redirect incoming requests (if possible), while never redirecting when rating lower than $n_i.A_{min}$. In all other cases, nodes with a tunable probability decide whether to serve or redirect.

In the previous scenario, if $n_2$ serves the redirected request, then the corresponding favor is marked as paid-back. Otherwise, $n_1$ may choose to use the corresponding TR as a means of accusing $n_2$ of acting selfishly. SeAl uses a DHT overlay of its own to store “complaints”. $n_1$ sends its TR to the appropriate node (say $n_3$) on this DHT (found by hashing the TR itself). $n_3$ then acts as an arbitrator between $n_1$ and $n_2$; it can ask (both) nodes to verify the TR and have $n_2$ pay back the corresponding favor. If the verification succeeds but $n_2$ still refuses to play fair, $n_3$ stores the TR for other nodes to know. If the verification fails, $n_3$ may choose to similarly “complain” about the perjurer peer.

What’s more interesting is that, if $n_2$ chooses to go altruist at some time, it can go out on the DHT, collect all filed complaints, and selectively pay them back, thus improving its status with respect to the community. Note that, while complaints are sent out to the DHT, TRs concerning favors done or paid-back are kept locally at the altruistic peer and presented along with requests.
originating from this peer, as a token of its contribution to the community. Moreover, to keep storage requirements constant as the system evolves, we use an aging scheme for stored TRs.

TRs are constructed using strong (e.g. public-key) cryptographic primitives, while nodes in SeAl are equipped with a public/private key-pair and identified using a digest of their public key, also used to verify TRs. Thus, nodes can’t fake TRs or refuse the validity of a TR, unless they change their ID (key-pair). Furthermore, SeAl deploys a feedback mechanism rather than a penalizing one – requests are queued and served in a prioritized manner, while the actual resources allocated for serving these requests (e.g. bandwidth, storage, etc.) vary, based on the overall “score” of the served peers. Moreover, peers commence their lifecycle in the system with the worst possible score, thus having no incentive to change their ID.

In conclusion, SeAl constitutes a novel infrastructure that addresses a key problem in P2P data sharing networks, namely the problem of wide-scale selfish behavior. Toward this goal SeAl offers (i) definitions/metrics of selfishness/altruism, (ii) subsystems performing monitoring/accounting and verification/auditing functionalities that enable the efficient, reliable, auditable identification of selfish peers, and (iii) accompanying incentive-offering mechanisms, while (i) respecting the autonomy of each peer to define his own selfishness/altruism levels and (ii) allowing for the exploitation of positive externalities (in the form of excess resources), which abound in P2P networks. Furthermore, depending on the environment in which SeAl is to be deployed, the modular architecture of SeAl permits the use of just SAL’s monitoring/accounting mechanisms, if we do not want to counter or do not expect to face malicious behavior, or to also use the extra security encompassed by SVL’s cryptographic verification mechanisms.

Our implementation and extensive performance testing of SeAl shows that SeAl achieves its identification goals swiftly. At the same time, the network, storage, and response time overheads imposed by SeAl have been measured to be very small, if any. SeAl can be the basis for the development of a wide variety of services in P2P data networks. As we shall see shortly, the ability to discover altruistic/powerful peers can play a key role in the derivation of better (more reliable and faster) network architectures and in the optimization of queries in p2p data networks. This is a major focus point for the work that follows.

2.4 Exploiting Altruistic Peers in P2P Data Networks

Our efforts in this thread have concentrated on the development of AESOP. AESOP intends to show how to leverage the coexistence of altruists and selfish peers found in real-life networks and harness them to improve routing performance. We focus on structured networks, and in particular we base our proposal on the desirable characteristics of Chord\(^1\), which include its simplicity, its acceptability within the community (as evidenced by the systems utilizing it, which include Limewire, MNet, Sun’s JXTA, and others) and its flexibility\([28]\). However, the proposed architecture can also be applied on other DHTs as well.

In general, in this work we define altruistic peers to be the peers that (i) stay connected for significantly longer periods of time, and (ii) are willing and possess the necessary capacity to accept greater loads. With these altruists’ characteristics in mind we revisit the “traditional” arguments about routing hot spots and about the overhead in dealing with the frequent topology changes inherent in P2P networks. Specifically:

- It is a good idea to concentrate most routing chores at altruistic peers; these peers are willing to carry extra load and have the required capabilities to do so. This results in more efficient routing than forcing weaker nodes to partake heavily in the routing tasks.
- The above decision will undoubtedly create greater routing tables at altruists. Traditionally, this causes greater reorganization overhead incurred when nodes enter and leave the network.

\(^1\)Due to this, the log()-notation in the examples to follow refers to base-2 logarithms.
However, the additional routing table entries of altruists will concern other altruistic peers. Because these stay connected for long periods of time, maintaining the freshness of this extra routing state does not result in prohibitively increased bandwidth overheads.

More specifically, our position behind AESOP is:

1. Weaving into the structured P2P network architectures the behavior and capability differences of peers, much-needed, quantifiable, and significant further routing speedups can be attained.

2. Routing speedups should refer to hop counts, routing state size and maintenance requirements, and robustness, and they should not be achieved by transferring overhead to other system operation phases (e.g., stabilization).

3. Routing speedups should pertain to the steady-state and highly-dynamic cases.

4. Altruistic and powerful nodes can be harnessed to offer these significant efficiency gains, while requiring that only a very small percentage of peers be altruistic, being burdened with only small overheads.

5. A software layer responsible for identifying and managing altruistic and powerful nodes can go long ways in offering these significant efficiency gains, while requiring that only a very small percentage of peers be altruistic, being burdened with only small overheads.

6. As a result, a paradigm that facilitates the cooperation of an altruist-based architecture and an auditing/accounting layer identifying altruist nodes is needed in order to take the next step in structured P2P network architectures.

2.4.1 State of the Art Overview

Looking at related research in DHT-structured P2P networks, one notices that, given a highly-dynamic environment, routing performance can degrade to $O(N)$ hops (that is, if the network remains connected). Fundamentally, this is due to the difficulty in keeping up with the required updates to routing state for special neighbors which ensure $O(\log(N))$ hops in the steady-state case. Much to their credit, the authors in [36] studied how to guarantee in highly-dynamic cases $O(\log(N))$ routing performance. To do this, $O(\log^2(N))$ so-called stabilization “rounds” need be ran by every node every half-life to update routing state (successors, predecessors, and fingers). However, this (i) transfers overhead from routing to the stabilization phases, (ii) this solution is expensive, yielding a total message overhead of $O(N \times \log(N))$ per half life: e.g., each node in an one-million node network needs to run on the order of 400 stabilization rounds, say, every half hour!, and (iii) detecting the presence/absence of low-bandwidth nodes (which are the great majority) during stabilization is time-consuming and highly error prone (think of nodes behind 56Kbits lines). Hence, given the huge scales and the highly-dynamic nature of the vast majority of peers, current architectures fail to ensure $O(\log(N))$ routing in the highly-dynamic case.

Furthermore, even $O(\log(N))$ hops, achieved in steady-state assuming “good node behavior”, may not be good enough: after all, these are overlay hops with each being translated into multiple physical network hops. In addition, even $O(\log(N))$ hops over peers with low bandwidth will definitely create performance problems. Finally, within the DHT world there is a complete lack of attention on exploiting powerful peers in order to improve performance.

But even when considering the unstructured P2P research efforts, one also notices a lack of considerable attention on research exploiting the heterogeneities among peer nodes[59]. As an exception, [53] talk about exploiting powerful nodes, which are thought of consisting of a number of smaller, “virtual” nodes. This transforms several hops among weaker nodes, into internal “virtual hops”
within a powerful peer. [38] present distributed algorithms to force-flow increased loads towards more capable nodes.

Still, heterogeneity means more than a mere distinction between powerful and weak nodes; there is also heterogeneity with respect to their behavior, being altruistic or selfish. For example, there will be powerful nodes that will not be acting altruistically. It is reasonable to expect that altruistic nodes will tend to have greater (processing, memory, bandwidth) capabilities, willing to share them (when not in use) with others (practically at very small extra costs, given a flat-rate resource pricing) – an expectation validated in [66].

Despite this, the aforementioned related work has made some good progress, showing the way in exploiting powerful nodes. Similar to our work, they are criticizing DHTs and structured overlays in failing to cope with highly dynamic environments, such as the ones expected in sharing P2P networks[14]. However, this led them to avoid using structured overlays, which unfortunately led to their inability to deliver definite performance guarantees, with respect to routing hop counts and robustness. Conversely, we follow a different path; we add further structure to DHTs, leveraging altruistic peers. In this way, we can deliver definite performance guarantees for the steady-case and, perhaps more importantly, for the highly-dynamic cases. Over and above any hop-count improvements, we ensure a more stable infrastructure, especially during high churn[55].

2.4.2 Our Contributions

The fundamentals of our approach are:

1. The $N$ network nodes are partitioned into DHT-structured clusters. For each cluster, a DHT-structured overlay network is created. Given the desired cluster size, $S$, and the number of clusters, $C$, overall, $C = \frac{N}{S}$ clusters are formed.

2. Each node requires minimal overhead for associating node IDs with cluster IDs (e.g. by hashing node IDs to cluster IDs).

3. The vast majority of a cluster’s peers are selfish. Within each cluster there exists at least one altruistic peer.

4. Altruistic peers maintain greater routing state, with an entry for all other altruists, creating a completely connected\(^2\) altruistic overlay network. Thus, communication between altruistic nodes (and thus between clusters) requires 1 hop.

5. Within every DHT cluster, all nodes maintain routing state for their neighbors, as required by the cluster’s DHT. Thus, each node has $O(\log S)$ neighbors. Also, all nodes keep routing state pointing to the altruistic node(s) in their cluster.

Routing is performed in two levels:

- Across clusters, from any node to a node in a different cluster: given the completely-connected altruistic overlay network, routing to reach any cluster from outside the cluster requires two overlay hops: one hop from a node to its altruist and another from this altruist to the altruist of the target cluster.

- Within clusters, from any node (including an altruistic node) to another node in the same cluster: routing is performed by sending the message over the cluster DHT network.

\(^2\)In practice, we talk about a “highly-connected” altruistic network, and $O(1)$ hops since complete connectivity is hard to achieve.
With AESOP we have presented a new paradigm for architecting structured P2P overlay networks, based on the coexistence of altruistic and selfish nodes in such environments. We have provided both analytical and experimental results of our architecture and have proven its superiority, compared to current approaches, with respect to routing efficiency, maintenance cost, routing state size, and robustness to failures.

In [42] we presented several architectural configurations and algorithms offering trade-offs between routing speedups vs the required number of altruists and their routing state and between routing path lengths in the steady-state case vs altruist-network connectivity requirements. The end result is that extremely small percentages of altruistic nodes are required, being burdened with small overheads, and introducing steady-state routing speedups by factors of up to $2-4 \times$, and by several orders of magnitude in the highly-dynamic case. At the same time, total routing state size is reduced by a factor of about $2 \times$, which leads to improved robustness.

Furthermore, routing robustness is improved due to the smaller total routing state and the isolation of the ill-effects of selfish behavior within small clusters of peers. Because of the above and its simplicity, we believe the proposed paradigm is viable and realizable and we offer it as the way to structure the P2P networks of the future.

3 Main Results on Information Dissemination in Internet-Scale Data Networks

3.1 The RangeGuard

3.1.1 The Problem

In this work [41] we address the issue of efficient processing of range queries in DHT-based peer-to-peer data networks. The novelty of the proposed approach lies on architectures, algorithms, and mechanisms for identifying and appropriately exploiting powerful nodes in peer-to-peer (P2P) data networks, the existence of which has been well documented in the literature.

DHTs provide us with strong probabilistic guarantees on the worst-case number of hops required to route a message from a node to any other node in the system. Unfortunately, all currently available DHT overlays are designed to only support single identity, exact-match queries. This has led researchers to investigate how they could enhance P2P systems to reply to more complex queries; however, up to now support for range query optimization remains an open problem. On the other hand, one of the main characteristics of widely deployed P2P networks (e.g. Gnutella, Mojonation, [69, 70] etc.) is that participating peers are largely heterogeneous, with regard to their processing power, available main memory and disk storage, network bandwidth, and internet connection uptime. Relevant studies of P2P networks have shown that this large heterogeneity is also depicted in the distribution of the query processing chores across the node population. Therefore, we believe that harnessing the power of such powerful and altruistic nodes is the key to providing efficient range query processing in a P2P setting.

3.1.2 State of the Art Overview

Related work in range query processing mainly propose architectures based on DHTs, such as Chord ([29], [64], etc), and CAN ([6], [20], etc). However, their efficiency is inferior compared with our proposed solution, either because they only provide approximate solutions ([29]), or because they provoke significant load imbalances [64], or because the underlying CAN routing protocol is expensive (in terms of hop counts) and less fault-tolerant, compared to Chord ([6], [20]). Karger and Ruhl [33] have presented a load balancing solution based on relaxing some of the constraints of Chord’s consistent hashing scheme: first, allowing every node to take one of $O(\log(N))$ positions on the Chord
ring \((N \text{ being the number of nodes})\), and then completely decoupling the placement of nodes on the
ring from any verifiable information. This latter decision is core to their approach to range query
processing. On the other hand, our architecture keeps the secure-hash-based placement of nodes
on the ring, while providing for efficient range query processing and storage/access load balancing.
Finally, as pointed out in [53], appropriately using powerful nodes in the location/routing primitives
of DHTs, can lead to both a more scalable and more efficient system.

3.1.3 Our Contributions

Our approach leverages existing DHT-based P2P research. It is based on:

1. extending the basic Chord architecture, using an order-preserving hash function and related
   algorithms for tuple addition/deletion and routing requests (query processing for range queries),
   and
2. a new architecture that facilitates the exploitation of powerful nodes in the network, assigning
   to them specific tasks for further significant speedups during range query processing.

This architecture is based on adding a second order-preserving Chord ring, composed of powerful
nodes, the RangeGuards, burdened with extra routing state and functionality above an initial order-
preserving P2P data ring containing all nodes of the network. RangeGuards take responsibility
(using consistent hashing) for arcs (ranges) of nodes on the lowlevel Chord ring. In addition, this
work defines a way to identify and collect RangeGuards, and presents mechanisms to utilize them
during range query processing.

Our analysis and experiments show that the proposed RangeGuard architecture can significantly
improve the efficiency of range query processing in terms of average hop counts. Furthermore, our
architectures can achieve good storage and access load distribution. Finally, and perhaps most impor-
tantly, a key advantage of the proposed architectures is that they avoid the dangers and inefficiencies
of relying on weak nodes for range query processing, with respect to their processing, storage, and
communication capacities, and their intermittent connectivity.

3.2 HotRoD: Replication Load Balancing and Range Queries in DHTs

3.2.1 The Problem

HotRoD is an architecture and related algorithms for studying the two problems of ensuring access
load balancing and efficiently processing range queries in a structured peer-to-peer (P2P) data net-
work. Placing consecutive data values in neighboring peers is frequently used in structured P2P
networks since it accelerates range query processing. However, such a placement is highly susceptible
to load imbalances, which are typically handled by either transferring or replicating data.

One of the biggest shortcomings of DHTs that has spurred considerable research is that they only
support exact-match queries. In case of range queries, each data value in a range should be queried
individually, which is greatly inefficient and thus infeasible in most cases. Much related research
deals with developing P2P systems that can support efficient range queries over DHT networks,
but their solutions suffer from data access load imbalances in the presence of skewed access (query)
distributions. There are a few solutions that handle the difficult problem of balancing data access
loads in the presence of range queries. These solutions are based on data transferring which is
inadequate in highly skewed access distributions, since transferring hot data items, simply transfers
the associated hot spots. Hence, in such cases, access load balancing is best addressed using the
replication of popular data values to equally distribute the access load among the peers storing these
replicas.
3.2.2 State of the Art Overview

There is significant work dealing with either range query processing, or load balancing in P2P networks. However, the great majority of related research deals only with one of these problems, avoiding the most difficult problem of balancing data access loads in the presence of range queries, while speeding up their processing. This combined problem is addressed in [23], [25] and [34], where load balancing is based on transferring load from peer to peer. We expect that this will prove inadequate in highly-skewed access distributions where some values may be so popular that single-handedly make the peer that stores them heavy. Simply transferring such hot values from peer to peer only transfers the problem. Related research in web proxies has testified to the need of replication [67]. Replication can also offer a number of important advantages, such as fault tolerance and high availability.

3.2.3 Our Contributions

We draw upon earlier work, [64] where we defined a novel order-preserving hash function and data placement algorithms to facilitate fast range query processing. The key contributions in [49] are algorithms for range query processing, load distribution, and replication management. The above can be applied to any DHT architecture, with only minor slight modifications. The proposed solution is tunable: by setting a parameter, the replication degree, we can tune between replication overhead costs and load imbalance. This is useful in cases we know, or can predict, the query workload. Our extensive experimental study of the proposed solution shows that with relatively small replication degrees, significant load balancing can be achieved, while continuing to provide efficient range query processing.

To our knowledge, this is the first work to concurrently address the issues of replication-based access load balance and efficient range query processing in structured P2P networks. Our main results show a significant hop count saving in range query processing from 5% to 80%, comparing with an enhanced version of Chord with different range query spans. Furthermore, access load is more smoothly distributed among peers: As an example, with a data replication overhead of about 100% we have found that the top 3% of the most-hit nodes receive 10% of the total load, compared to 60% of the total load they would receive without HotRoD. At the same time, the load is transferred to less-hit nodes some of which receive only a small load increase compared to a perfectly uniform load distribution. Also the further the range query spans, or the access skewness increases, the benefits of our solution increases.

3.3 The Publish/Subscribe Paradigm over DHTs

3.3.1 The Problem

The peer-to-peer (P2P) paradigm is appropriate for building large-scale distributed systems/applications. P2P systems are completely decentralized, scalable, and self-organizing. A large body of research is currently targeting the extension and employment of DHTs for efficient data query processing. The nature and functionality of the DHT-based P2P can guarantee the efficient managing of queries with exact-match equality predicates. However, it is difficult to perform queries with range predicates over numeric attributes and/or prefix, suffix, and containment predicates over string attributes.

Our work on this area extends the functionality of traditional DHT-based P2P networks. We first provide our solution that efficiently supports range predicates (=, <, >, etc.) on numeric values in DHT-based systems [63]. The next step is to support a rich set of predicates on string attributes such as equality, suffix, prefix, and containment, [4]. Specifically, supporting queries over string attributes over a DHT infrastructure is an open problem.
We primarily focus on the pub/sub technology and formulate our solution in terms of a pub/sub infrastructure build on top of a DHT-based P2P network. Our solution is independent of the type of DHT and can be easily applicable to every DHT that can efficiently locate an object based on its key identifier.

3.3.2 State of the Art Overview

The richest more powerful publish/subscribe systems are the so-called *content-based* pub/sub systems. Content-based systems give users the ability to express their interest by issuing continuous queries, termed *subscriptions*, specifying predicates over the values of a number of well defined attributes. Events are published into the system signifying the occurrence of a real-world event (such as the addition of a data item, the execution of a stock-exchange transaction, etc.) The matching of publications (events) to subscriptions (interests) is done based on the content (values of attributes).

Recently, some attempts on distributed content-based publish/subscribe systems use routing trees to disseminate the events to interested users based on multicast techniques [7], [10], [62], [46]. Some other attempts use the notion of rendezvous nodes, which ensure that events and subscriptions meet in the system [47]. However, none of the published research has managed to show how to leverage DHTs as a dominating technology for constructing efficient scalable overlay networks in creating a content-based pub/sub infrastructure with extended functionality.

3.3.3 Our Contributions

The main challenge in building a pub/sub system in a large-scale distributed environment is the development of an efficient distributed matching algorithm. In a pub/sub system an event is said to match a subscription if and only if all the subscription’s attribute constraints are satisfied. Each subscription has a k-bit identifier, SubID. The main idea behind our solution is to store SubIDs for each subscription in appropriate nodes over a DHT network so to have incoming events check those nodes for potential matching and to leverage the logarithmic performance guarantees of DHTs so to ensure scalable event matching performance.

When dealing with string attributes our solution supports prefix, suffix, containment, and equality constraints. The main idea of our approach is to store the subscription ids (SubIDs) at those nodes of the DHT network that were selected by appropriately hashing the values of the attributes in the subscriptions. In each node there are three different lists of SubIDs based on the kind of constraint (prefix, suffix, and equality. The containment can be easily transformed to prefix and suffix operation).

For each string attribute of the incoming event we ask all possible nodes for their stored SubIDs whose ids are derived by hashing all possible prefix or suffix sub-strings of the string value. Next, we check its SubID in the collected lists and based on the information we have about the number of attributes each subscription has (a piece of information contained in the SubID identifier) we identify the matched subscriptions.

During the procedure for processing incoming subscriptions, our performance analysis has shown that in a DHT of N nodes we need to contact \( O(\log(N)) \) nodes in the worst case for each attribute of the subscription. The event-matching process requires to contact \( O(c \times \log(N)) \) nodes for each event, where \( c \) is a constant which depends on the number of the event’s attributes and on the string-length of each attribute’s value.

We also contributed several distributed event-matching algorithms offering trade-offs between mainly the processing cost of collecting and performing the matching at a single node versus the overall network traffic to transfer data. Our motivation is to distribute when possible/profitable the matching phase to a number of involved DHT nodes, without incurring high bandwidth overheads.
In conclusion, in this thread of work [63, 4] we have shown how to leverage DHT-based P2P systems, towards building scalable, self-organizing, well-performing systems that support queries with a rich set of constraints on string and numeric attributes. We specifically focused on and presented how our algorithms can be applied in a publish/subscribe environment with a broker network implemented using a DHT. The proposed solution is DHT-independent and can be applied in every DHT infrastructure that provides the basic functionality of finding and reaching the node that stores an object with a specific key value. To our knowledge, this is the first work that shows how string attribute queries (with equality, prefix, suffix, and containment predicates) can be processed over a DHT infrastructure.

4 Result Dissemination

The research conducted in the context of this WP has led to 9 publications [4, 44, 42, 43, 41, 49, 63, 64, 65] in international conferences and refereed workshops, and another 3 papers [5, 45, 48] are under submission for publication. The publication list includes 3 full papers in prestigious, highly selective, first-rate conferences; specifically, ICDE [44], CIKM [4], and EDBT [49], all of which have acceptance ratios of 17% or lower.

References


[14] Chawathe Y. et al., "Making Gnutella-like P2P systems scalable", in Proc. SIGCOMM '03.

[15] Clarke I. et al., "Freenet: A distributed anonymous information storage and retrieval system", in Proc. PET '00.


[28] Gummadi K. et al., "The impact of DHT routing geometry on routing resilience and proximity", in Proc. SIGCOMM '03.


[31] Huebsch R., J.M. Hellerstein, N.L. Boon, T. Loo, S. Shenker, and I. Stoica, "Querying the internet with PIER", in Proc. VLDB '03.
[38] Lv Q., S. Ratnasamy, and S. Shenker, "Can Heterogeneity Make Gnutella Scalable?”, in Proc. IPTPS ’02.
[40] Ngan T., D. Wallach, and P. Druschel, "Enforcing fair sharing of peer-to-peer resources”, in Proc. IPTPS ’03.


[59] Saroiu S., P. Gummadi, and S. Gribble, "A measurement study of peer-to-peer file sharing systems", in Proc. MMCN '02.


[66] Wilcox-O’Hearn B., ”Experiences Deploying A Large-Scale Emergent Network”, in Proc. IPTPS '02.


