Network-Aware DHT-Based P2P Systems

Marguerite Fayçal and Ahmed Serhrouchni

Abstract P2P networks lay over existing IP networks and infrastructure. This chapter investigates the relation between both layers, details the motivations for network awareness in P2P systems, and elucidates the requirements P2P systems have to meet for efficient network awareness. Since new P2P systems are mostly based on DHTs, we also present and analyse DHT-based architectures. And after a brief presentation of different existing network-awareness solutions, the chapter goes on effective cooperation between P2P traffic and network providers' business agreements, and introduces emerging DHT-based P2P systems that are network aware through a semantic defined for resource sharing. These new systems ensure also a certain context-awareness. So, they are analyzed and compared before an open end on prospects of network awareness in P2P systems.

1 Motivations

Peer-to-peer (P2P) networks are an evolution of the Internet network, and lay over it. However, the Internet network is composed of tens of thousands smaller independent networks called Autonomous Systems (AS) and belonging to various administrative entities (e.g., network providers, universities, companies, etc.), each having its own routing policies. Bilateral connections between ASs are governed by business agreements negotiated between the entities they belong to [1]. And routing information between ASs is exchanged via an exterior gateway protocol such as BGP

Marguerite Fayçal
Institut Telecom, TELECOM-ParisTech, 46 rue Barrault, 75013 Paris, France,
e-mail: Marguerite.Faycal@telecom-paristech.fr

Ahmed Serhrouchni
Institut Telecom, TELECOM-ParisTech, 46 rue Barrault, 75013 Paris, France,
e-mail: Ahmed.Serhrouchni@telecom-paristech.fr

X. Shen et al. (eds.), *Handbook of Peer-to-Peer Networking,*
DOI 10.1007/978-0-387-09751-0_12,

[2]. But in P2P networks, routing tasks are distributed across all system peers in an autonomous and spontaneous way without taking the IP network structure or routing into account, and consequent traffic often cross network boundaries multiple times [3], causing redundant traffic and extra delay, and overloading links, increasing the risk of their congestion.

Last years, many researches went on to investigate and understand the relation between routing in the overlay and underlay IP network. Reference [4] shows how current ISP traffic engineering techniques are inadequate to deal with emerging overlay network services, as overlays do not follow ISP's policies. It also illustrate how an uncoordinated effort of the two layers to recover from failures may cause performance degradation for both overlay and non-overlay traffic. The problem of rerouting around failed links was also tackled in [5], which finds out that tuning underlay routing parameters improves overlay performance. Using game theoretic models, [6] studies the interaction between overlay routing and traffic engineering within an AS, and shows that the selfish behavior of an overlay can cause huge cost increases to the whole network. Also [7], which deals with voice over IP as a successful application of P2P, shows that the quality of the relay paths could be improved when the underlying network AS topology is considered.

Moreover, focusing on P2P platforms and applications, their overall performance also depends on the performance of their background P2P routing protocol. So, new systems are mostly based on distributed hash tables (DHT), which are algorithms that provide efficient mechanisms for resource location. However, DHTs assume the system is uniform in available resources and that every node participating in the DHT is within the same transport domain.

Otherwise, despite various hurdles, as copyright issues, legality matters, and security concerns, P2P networks and systems still gain in popularity. The research community continues also to develop interesting applications of P2P technology, together with new platforms for application development. P2P is then used for increasing scalability and decreasing the cost of management and deployment. The consequently continually growth of the P2P traffic appears thus somehow daunting for network providers. An effective cooperation between both parts is then challenging, so that according to the underlying network, peers can find the best instance of the resource they target. But underlay awareness at the P2P level becomes therefore necessary and unavoidable.

2 DHT-Based Architectures

Mathematically, a *DHT* is a distributed injective hash function that associates keys with values, both in the same logical key space *K*. In DHT-based structured P2P systems, the ownership of this key space is split among the active peers of the network. A global unique identifier is thus assigned to every node; it is known as the *nodeID*. Similarly every object has a global unique *objectID*. An object can be any kind of resource.

DHT algorithms map then *objectIDs* to the node responsible for that object. They implements lookup and retrieval functionalities providing two main functions: a *put(key, data)* function that fills the table with couples of corresponding keys and values, and a *get(key)* function that permits to locate an object, taking the object's *key* (which is the *objectID*) and returning the *nodeID* of the peer responsible for that object [8]. The responsible node then either supplies the object directly or indicates where (or how) it can be acquired.

So, given a hash function h that should balance the distribution of keys throughout the logical space K, and a resource identifier r that the peer p of address IP $@IP$ wants to share on the network, we have then:

$$h(r) = k \in K \text{ ; where } k \text{ is the } objectID$$
$$\text{and}$$
$$h(@IP) = i \in K; \text{ where } i \text{ is the } nodeID$$

Then, the resource r, or a link towards this resource, is stored on the peer p of address $@IP$, so that *dist (k, i)* is minimal; *dist (k, i)* being the distance between k and i, according to the distance definition in the logical space K. In other words, each peer owns all the resources (or links towards the resources) whose key is, in the logical key space, closest to the identifier of that peer. Thus, in the DHT, k is mapped to i, and the peer p can retrieve r. Likewise, each peer identified by j can retrieve, thanks to his DHT, any resource whose identifier m is closest to j. If j doesn't have m in its DHT, it has at least an n closest to m. Consequently, each key lookup is resolved by iteration, in multiple steps, resulting in a multihop path to be taken in the overlay [9]. DHTs can thus effectively route messages to the unique owner of any given key, and they are typically designed to scale to large numbers of nodes. Typically, the worst case cost to locate an object, in terms of number of messages exchanged, is logarithmic with the number of peers.

The key feature of DHTs used to build P2P architectures is that they use consistent hashing [10], which means that changes to the location where data items are stored are minimal when new locations are added or old locations are removed. In fact, as peers enter and leave the network, messages are exchanged between the peers in the DHT to preserve the structure of the DHT and exchange stored entries. DHTs provide thus data with high degree of availability across a set of peers with dynamic membership. Various DHT implementations may visualize the hash space as a line, a ring, a tree-like structure, a grid, etc. Reference [11] discusses basic geometry underlying DHT routing algorithms and how it impacts their performance in two important areas: static resilience and proximity routing.

All DHT-based P2P architectures share the following four main properties [12]:

1. *Low degree*: each peer keeps only a small number of active connections to other peers.
2. *Low diameter*: the maximal number of necessary hops to reach any peer of the network is minimized.
3. *Greedy routing*: peers independently calculate a short path to the destination.
4. *Robustness*: even if links or peers fail, a path to the destination can be found.

In DHT-based P2P systems, nodes organize themselves in accordance with the DHT's implementation to form a communication graph with an optimal diameter / degree trade-off. Each peer maintains addresses of other peers participating in the same DHT in a routing table, sorted according to specific criteria. And in order to reduce path latency of distributed queries, DHT algorithms include round trip time estimations among such criteria [13].

Nevertheless, DHTs suffer some limitations. First of all, the partitioning of the key space in a DHT is directly related to the number of nodes residing in the system at any given time. As such, whenever a new node joins the system or an existing node leaves it or fails, the partitioning changes, and data must be moved so that the system still works properly and live nodes still be able to determine through the DHT substrate the current live node responsible for the key they look for. Second, DHTs are not designed to answer interval queries, since they link only one object to each key. Lastly, the problem we address in this chapter is the fact that DHTs leave aside the structural aspect of the underlay IP network, assuming that every node participating in the DHT is within the same transport domain. In fact, a single hop in the DHT is likely to involve multiple routing hops in the Internet, and successive hops may lead a message travel back and forth several times. Thus, the physical path so travelled is often less than optimal. DHTs assume also that the system is uniform in resources, since they leave also aside the available resources at each node, such as network bandwidth, free processor, or storage capacity available at each active peer.

More details on DHTs' evaluation can be found in the following. Reference [14] addresses problems in DHT P2P applications. Reference [15] discusses churn. And [16, 17] present performance studies of various DHT algorithms with and without churn.

DHT-based P2P routing protocols are commonly called DHTs. They were first introduced to the research community through four different architectures: Chord [18], Pastry [19], CAN [20], Tapestry [21]. Since that time, a plethora of other DHTs emerged. Some of the most well-known ones are Kademlia [22], Viceroy [23], Bamboo [15], D2B [24]. And the list still grows longer, although very few are publicly released with robust implementations. However, Kademlia is already integrated in two of the most popular P2P filesharing applications: BitTorrent [25], Emule [26].

3 Requirements

As a direct consequence of the total distribution of the Internet, and as DHT-based networks lay over the Internet, it is quite likely that topology information would be distributed at an ISP, a network provider, or an AS or domain level. Thus, beside scalability and robustness, network-aware P2P systems aim to satisfy both P2P users and ISPs. The former are interested in an enhanced time of data retrieval, with better performance; the latter are interested in avoiding congestion on critical

links, and look forward to both the optimization of network resources usage, such as bandwidth, and the reduction of operation costs.

Effectively using the underlay network information implies two essential parts:

- firstly, discovery techniques, generating and providing underlay network proximity information,
- secondly, techniques exploiting such information in both query routing and data retrieval processes.

Three techniques of generating proximity information may be identified [27]:

- *Expanding ring search* is a flooding technique for measuring the round-trip time (RTT) between a node and all the others within a defined radius (in terms of network hops).
- *Heuristic based approaches* consist in measuring the RTT between a node and its close neighbours only.
- *Landmark clustering measures* RTTs to selected landmark nodes, and sorts the landmarks in terms of increasing RTTs. Intuitively, nodes having similar distances to these landmarks are close to each other in terms of network latency. This scheme is used in [28].

Vivaldi [29] assigns synthetic coordinates to Internet hosts, so that the Euclidean distance between two hosts' coordinates predicts the network latency between them, with no need of landmark nodes.

Most works on estimating topology information focus on predicting network distance in terms of latency. But for many P2P applications, e.g., video-based ones, throughput is often a more important quality metric. The *iPlane* service [30] aims then to generate and maintain an "atlas" of the Internet using active measurements that contains information about latency, bandwidth, capacity and loss rates between arbitrary Internet hosts.

However, available bandwidth and lossrate estimation from end hosts may be somewhat obscured by lastmile bottlenecks. The notion of network tomography summarizes techniques of active network probing and passive traffic monitoring to infer information about the network topology and link-level characteristics [31].

Lastly, the topology awareness of overlay networks can be modelled, involving a mathematical metric for the degree of topology matching between an overlay network and its underlying physical network [32]. The model is based on an optimization problem, a NP hard problem but solvable in polynomial time for some particular inputs [32].

4 State of the Art

First DHT-based P2P protocols were developed agnostic of the underlay topology. However, Pastry [19] uses certain heuristics to exploit physical network proximity in its overlay routing tables. But decisions are only made when there is

a choice between nodes, and locality is not exploited for the underlying routing strategy.

Existing techniques that permit to exploit the network proximity information are subdivided into three categories [33]: proximity routing, geographic layout or topology-based *nodeID* assignment, and proximity neighbour selection [34].

1. With Proximity routing, the overlay network is built independently of the physical topology. During the routing process, if a peer has the opportunity to choose among k possible next hops, it chooses to route the message towards the closest node in the underlay network, among this set of k nodes. An alternative is to choose to route towards the node that represents the best compromise between proximity and progress in the overlay key space. The overall performance of this technique depends thus on k that is proportional to the size of the routing table. Moreover, small hops may be thwarted by an increase of the number of hops.

2. Topology-based *nodeID* assignment aims to map the overlay key space onto the underlay topology, and biases the *nodeID* assignment. Consequently, delays decrease certainly, but uniform distribution of *nodeIDs* in the overlay key space is violated, which leads to loadbalancing problems. Neighbouring nodes are also likely to suffer correlated failures, and thus both robustness and security of the system are likely to be weakened. However, this technique has been successfully used to create a topologically sensitive CAN [28] but it cannot be applied in a one-dimensional id space (e.g., Chord [18], Pastry [19]).

3. Proximity neighbor selection: Like the previous one, this technique builds a topology-aware overlay. But instead of biasing the *nodeID* assignment, it chooses the routing table entries to refer to the closest nodes at the underlay level among all the ones that satisfy the algorithm. This technique is likely to have success only when applied to an overlay protocol that allows certain freedom in constructing routing tables without affecting the diameter. Thus, it can be used with Pastry [19] when constructing the neighbourhood set, but not with CAN [20] or Chord [18], where each routing table entry refers to a very specific point in the overlay key space. Consequently, proximity neighbour selection introduces only low overhead when implemented, and facilitates the cache management since overlay neighbours are likely to be also neighbours at the underlay layer. It can also be viewed as a good compromise decreasing the network's diameter and preserving the load balance, but neighbours' discovery is still tightly related to the protocol.

These three techniques of using generated network proximity information have their limits and are enormously linked in the overlay protocol. Thus, new structural solutions emerged afterwards, and the list still continues to lengthen.

Hereafter, we give a brief presentation of different well-known DHT-based topology-aware P2P systems, namely: Brocade [35], the *expressway* [36], Hieras [37], Toplus [38] and Plethora [39].

Brocade [35] adds a secondary overlay on top of a primary DHT-based P2P network that exploits knowledge of the underlying network characteristics. The secondary layer consists of nodes having high network capacities (bandwidth, processing power), and situated near the network access points such as gateways and routers. These nodes are called supernodes and act as landmarks for each network domain. Each supernode manages a set of local nodes to reduce the traffic in the network, but has to keep information about all overlay nodes inside its domain. They may thus become bottleneck points. To route a message each node first connects to the nearby supernode, then uses the second layer as a shortcut to the destination, reducing both network bandwidth usage and overall physical hops. But this layer still uses a logical routing, which involves for each logical hop several ones at the underlying IP level. Thus, to a certain extent, Brocade pushes the problem back to a secondary network of smaller size.

The *expressway* [36] is an auxiliary network constructed on top of any primary DHT-based P2P overlay, in order to take advantage of the inherent heterogeneity of the underlay network (node connectivity, physical proximity, forwarding capacity, node availability). Two generic approaches exist for such a construction. The first approach uses the AS-level topology derived from BGP [2] reports. The second approach uses a landmark numbering technique that enables proximity neighbour selection and can deal with changing network conditions. The constructed auxiliary network is called an expressway as it is intended to speed up routing. It is formed by nodes with high network capacity, situated near gateways or routers, and called expressway nodes. The first approach requires that every node knows all the nodes in its AS. And to route a message, a node contacts first its local expressway node: the one in its AS or its landmark cluster. In the second one, routing is similar to Distance Vector Routing.

Hieras [37] is a multilayer hierarchical system intending to relieve the problem of distributed overlay routing tasks without awareness of underlay network link latency. At the highest level, peers are grouped in one big ring, and at the lower ones, topologically adjacent peers are grouped into several disjointed rings. But each peer belongs to all the levels simultaneously and manages thus more than one successor list. Each ring is a subset of the overall P2P network and is created in such a strategy that the average link latency between two peers in lower level rings is much smaller than higher level rings. To estimate such proximity for each ring, Hieras employs distributed binning [28], requiring thus the existence of well-defined landmarks nodes. In Hieras, routing tasks are first executed in the smallest ring first. If it fails, it moves up to higher level rings, until eventually reaching the highest level. But a majority of routing hops previously executed in the global P2P ring is so replaced by hops in lower level rings, reducing thus routing latencies.

Toplus [38] is a hierarchical lookup service for structured P2P networks. It organizes peers in groups according to their network IP prefixes, and groups into a new higher order group, and so on following the Internet hierarchical topology it gets from BGP tables [2]. Thus, an AS is subdivided into an IP hierarchy. The routing mechanism of Toplus is based on a generalization of longest prefix matching of IP addresses, using the XOR metric, like Kademlia [22]. Both structure and look up

performance of Toplus follow those of the Internet network. However, Toplus suffers a number of drawbacks. Obviously, it is sensitive to correlated node failures that may bring down an entire set of related IP addresses. And another matter is the unbalanced load among the nodes, since the population of the id space is not uniform, because the number of active nodes in an inner group is not necessarily proportional to the IP addresses it covers.

Plethora [39] is a two-layer wide area read-write repository, where peers are expected to have a partially persistent network state and good Internet connectivity in terms of bandwidth. On the top of the global overlay that contains all the peers, the Plethora routing core organizes peers into several local overlays according to ASs and associated proximity. Local overlays serve as locality-aware caches for the global overlay to improve access time to data items. Queries are thus routed first in the local overlay. The size of the local overlay impacts the system's performance and is thus defined by two system parameters: a maximum and a minimum numbers of peers. The size of each local overlay is controlled by the one of its peers with the smallest identifier. Two distributed algorithms are also active in the system. One for merging local overlays when necessary and another one for splitting them with a high probability guarantee that nodes in the same AS stay in the same local overlay after a split operation.

Many other systems still emerge to enhance P2P system's performance by exploiting underlay topology information. And recently, new architectures emerged for topology estimation through layer cooperation, allowing cooperation between P2P traffic and network providers' policies.

A simple scheme [40] can let P2P overlay interacts with underlying ISP infrastructure. It relies on a server, called the oracle, hosted by the ISP and that helps P2P users choose optimal neighbours. More precisely, a P2P user sends the list of potential neighbouring peers to the oracle, which ranks this list based on a number of factors that each ISP can decide individually, like their proximity to the user or higher bandwidth links, or according to its routing policies or its agreements signed with other ISPs. The oracle acts then like an abstract routing underlay to the overlay network but as it is a service offered by the ISP. It has thus direct access to the relevant information (e.g., the one concerning the topology) and does not have to infer or measure it. Although the oracle can be deployed with DHTs, reference [40] focuses mostly on unstructured P2P systems.

Provider Portal for P2P (P4P) [41] is a light-weight architecture enabling explicit communication between P2P (independently of DHTs) and network providers, in order to reduce backbone traffic and lower operation costs. The proposal leverages the fact that the ISP is best-positioned to determine locality and to direct clients not only to nearby peers but also to peers that are accessible over well-provisioned and lightly loaded links. P4P framework consists of a data-plane component and a control-plane component with an *iTracker* providing three kinds of information regarding the network provider: network status/topology, provider guidelines/policies, and network capabilities. However, with the *iTracker*, the P4P framework seems to be a kind of centralized architecture applying CDN (Content Delivery Network) architecture to a filesharing P2P network.

5 Semantics for Resource Sharing

Semantics in P2P systems is an active area research, but it focuses on the problem of mapping or retrieving semantically close data shared by different peers. This paragraph does not deal with the resources' semantic, but tries to define semantics for the identifiers in a DHT, based on the underlay topology in order to lead to a win–win situation where both P2P users and network providers meet their requirements. Hereafter we present two such systems, namely CAP (a Context-Aware P2P system) [42] and NETPOPPS (a NETwork Provider Oriented P2P System) [43]. Both proposals are multi-layer systems exploiting the injective cryptographic hash function of the DHT, which is also used to build the different identifiers of the P2P system. Following proposals are thus independent of the P2P algorithm implemented at the primary global overlay.

5.1 A Context-Aware P2P System

CAP [42] aims to build context-aware *nodeIDs* and *objectIDs*. Therefore it proposes to compute the different identifiers of the system at the secondary levels using HMAC (Hash based Message Authentication Codes) [44], a keyedhashing function, originally aimed for message authentication, thanks to a secret key it uses.

HMAC is a message authentication code that uses a cryptographic key in conjunction with a hash function in order to guarantee integrity between a sender and a receiver. It is computed as follows.

$$HMAC(h,k,m) = h(k \oplus opad || h(k \oplus ipad || m))$$

Here are details of the equation.

- h is a commonly used cryptographic hash function; in CAP it is the one used by the DHT of the global P2P overlay.
- k is initially defined as a secret key, but in CAP it is a configurable system parameter, called HKey, and which value is known to all the peers of one DHT. Thus, a HKey labels each DHT, which is then known as VDHT (for Virtual DHT). A detailed definition is given in the next paragraph.
- m represents the text to be hashed; it is thus either the object name or the address IP of the peer.
- $ipad$ and $opad$ are the bytes 0x36 and 0x5C respectively, each repeated 64 times.
- $\oplus$ denotes the bitwise XOR operation.
- $||$ denotes the concatenation of two bit strings.

Coming back to the HKey, it could be a simple or compound one. A simple HKey is based on one parameter or a single criterion: it could be a defined characteristic (network metric or policy). A compound HKey is based on the combination of any two or more characteristics (parameters or criteria) and is computed as the output of

the cryptographic hash function used by HMAC and applied to the concatenation of those characteristics; e.g., if we are interested in searching a resource according to three metrics $m1$, $m2$ and $m3$, then $HKey = h(m1||m2||m3)$.

A compound HKey could be derived in two or more, simple or compound HKeys: these derived HKeys are then typically based on one or more different characteristics from those the initial compound HKey is based on. But derived HKeys could also be defined with simple HKeys; e.g., if a certain simple HKey could take four values for the same peer (e.g., if the peer resides in four zones), derived HKeys could be defined as a combination of two or more of those values.

The type of the HKey is defined in a global system profile. (The way the system profile is loaded in the system does not affect node operations or routing mechanism; e.g., it could be loaded on a kind of bootstrap node managed by the system operator or service provider.) So, in case of a derived HKey, the profile defines which combination of two or more values or characteristics to take into account, and eventually in which order. In fact, the profile defines a set of priority criteria and identifies the parameters to consider in the routing protocol, defining thus the HKey structure. These criteria and parameters are quantified as the peer arrives in the system, according to an external procedure that doesn't affect the routing mechanism and that could be based on some database reports of the system (e.g., the BGP reports [2] for the identifier of the AS the peer resides in).

Examples of what a simple HKey could be are: the AS identifier, the administrative domain name or the group name or identifier of a specific group communication or a specific secure group, a QoS parameter (e.g., minimum available bandwidth, minimum battery power for mobile systems, minimum storage capacity, etc.), a location parameter (e.g., network or real distance, GPS location, country, etc.), a type of shared files (e.g., a specific movie or audio file type), a language or a topic of shared files (based on metadata), a secret key, a keyword, etc. A compound HKey could be based on two or more such simple HKeys. Derived HKeys could then be defined depending on the routing policy of the system, defined by the above mentioned profile.

Consequently, a heterogeneous P2P network can be layered in a set of different uniform overlays, each one called a zone and characterized by a specific HKey. In order to have homogeneous identifiers in the whole system, each zone is identified by $h(HKey)$. The HKey will then also label both an overlay and the DHT in its corresponding key space, above named VDHT.

Each node can participate in one, two or more of these overlays, depending on its properties (available resources, locality, group membership, etc.) and according to the semantics of the HKey that is taken into account, but resides in at least one overlay, the global one, which could for example be at the lowest overlay level. The other overlays are called local ones, and at one level there could be many zones. In case the system profile does not define derived HKeys, there will be only a single secondary overlay at the top of the global one; else, there will be L more local overlays, where L will be equal to the maximum number of the different possible defined derived HKeys per peer.

The routing mechanism is the same at each level according to the P2P routing protocol defined at the global overlay; the only difference is the HKey characterising the level and thus the usage of the consequent different *nodeIDs*, *objectIDs*, and DHTs.

When a peer needs a data item, it first searches the local overlay it resides in. If the query fails, the peer will then searches the global overlay where the identifiers remain unchanged. When derived HKeys are defined, before searching the global overlay, a peer searches each zone it resides in. To improve the data retrieval performance in the system, a data item is automatically cached in the requester's local overlay if it is retrieved from the global overlay (and of course, its objectID will be then computed using *HMAC* and based on the HKey corresponding to that local overlay).

The joining and failure/departure operations are the same at each level according to the routing protocol implemented in the system; but after a node joins the system (at the global overlay) and before it joins any local overlay at any secondary level, some operations must be taken in order to join the secondary overlay if it exists or to build it if possible.

Otherwise, each existing zone is associated with a data table, called zone table, having the same identifier, and stored on a rendezvous point of the global overlay. The zone table contains up to four *nodeIDs* in the global overlay of nodes already participating in its corresponding local overlay. Thus, a new node in the global overlay has first to look up $h(HKey)$ to be able to join a zone with its corresponding $nodeId$.

If a new node is the first one asking to join its local overlay, then it creates a zone table with the identifier of that local overlay and its own global *nodeID*, it stores the created data table at the rendezvous node, and it starts the local overlay.

CAP guarantees that a data item will be retrieved from the zone where it has been found. In fact, at each secondary layer, when an overlay is populated, all nodes and objects have their identifiers based on the HKey characterising the overlay; so if a query initiated by a node residing in the overlay does not fail, that means that the response is present in this overlay. And even if the response is a pointer to the requested object, the object is necessary in the overlay, since every data presented or represented in a secondary overlay has its identifier necessarily computed based on the HKey characterising that overlay.

Consequently, CAP ensures through a specific semantics that the result of a context-oriented query is context-oriented.

5.2 A Network Provider Oriented P2P System

To enable tight cooperation between P2P traffic and network providers' routing policies and business agreements, NETPOPPS [43] layers P2P overlays in a set of different uniform overlays, and applies the principle of key derivation to *nodeIDs*. But

an object keeps a unique identifier at all levels, namely its primary objectID, the identifier it has at the global P2P layer. In fact, objects are managed by nodes.

The principle of key derivation is an efficient key management technique that was proposed in [45] to satisfy the needs of private hierarchical group communication requiring specific confidentiality upward and downward the hierarchy. It is illustrated by Fig. 1, where h is a distributed injective hash function. Considering any node of *Level 1* identified by $Id_1 = k$, it will have at *Level 2* an identifier $Id_2 = h(Id_1)$. This computational relation applies for the identifiers of a same node between any two successive layers, i.e., $Id_i = h(Id_{i-1}) = h^{i-1}(Id_1)$ at any level i. So, any node of level i can compute its identifier Id_j at a level j, where j is greater than i, by deriving $Id_i(j-i)$ times; in other words: $Id_j = h^{j-i}(Id_i)$.

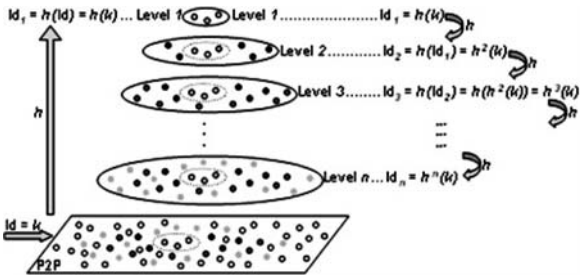

Fig. 1 Principle of key derivation for key management

Regarding the system architecture, as illustrated by Fig. 2, secondary overlays are characterized by the width of their constituent entities in term of number of basic entities (BE) merged together. A BE is any kind of routing domain managed by a Network Provider (NP). NETPOPPS proposes that a P2P path connection emerging from any peer of an entity remains entirely inside the same entity. It also proposes that according to NP's business agreements two BEs merge in an intermediate entity (IE) to allow a P2P path connection emerging from one of them to end in the other one. Two IEs can also merge in a new IE, and so on. Besides, secondary overlays lay on the top of the primary P2P overlay in descending order of the width of their entities, in terms of number of merged BEs. This makes the system looks like a collection of several hierarchical subsystems composed of a BE on the top of as many IEs and managed by NPs; each NP managing as many subsystems as BEs it provides. The identifier of an entity is called a *vector* and labels both the virtual overlay in the corresponding entity and the DHT in its corresponding key space. The *vector* of an IE is in fact built in a vector format, by listing in an ascending order the identifiers of all its constituent BEs.

The routing mechanism in each overlay is the same as in the global P2P network; only the *nodeIDs* and DHTs in use are different. When a peer needs a data item, it first searches the local overlay (of the BE) it resides in. If the query fails, the peer will then similarly searches for the data item in the overlay corresponding to

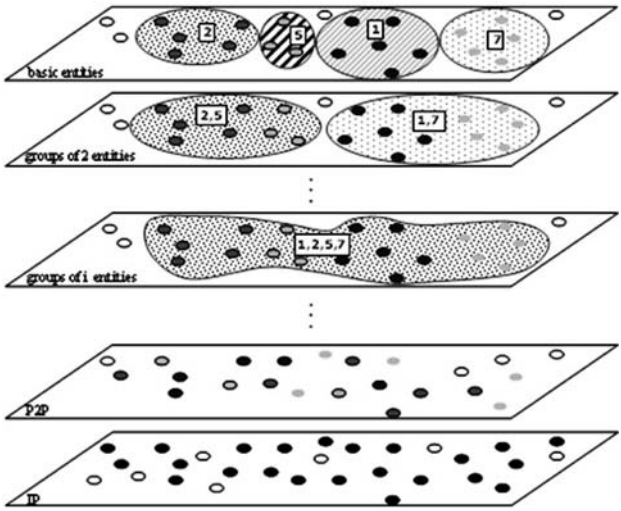

Fig. 2 Overview of NETPOPPS

the first IE; and so on, the peer continues searching for the desired data item in each intermediate overlay going down the hierarchical subsystem, until success. But during a single routing process, only the originator of a query can switch from an overlay to another. The node requester also automatically caches in its local overlay a data item it retrieves from the global or an intermediate overlay.

The joining and failure/departure operations of nodes are the same at each level according to the routing protocol implemented in the system. However, after joining the global overlay and before it joins any other secondary overlay, a new node has to identify the secondary overlays it has to join. NETPOPPS proposes thus that each NP manages a single control node (CN) per BE. The CN does not participate in the routing procedure. It holds a profile that memorizes the vectors of the different IEs of the hierarchy going down from its BE. The NP can also decide on other features to deploy on the CN and other data to save in the profile.

Like any new joining node, the CN joins first the global overlay. Then, as illustrated by Fig. 3, the CN computes its different identifiers according to the principle of key derivation, except an additional derivation: the first derivation of its primary *nodeID* gives its *nodeID* in the BE's overlay. Consecutively, the CN initiates the dif-

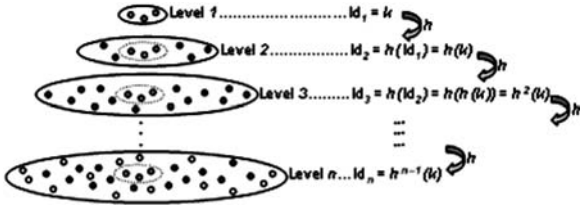

Fig. 3 Computation of the different identifiers of a new joining node

Table 1 Structure of the reference table

vector of the BE	primary ID of CN	vector of the 1st IE	...	vector of the Nth IE

ferent overlays. Afterwards, it shares at the global overlay a Reference Table (RT), represented by Table 1, and identified by $h(vector\ of\ BE)$.

So, to correctly join the different secondary levels, a new node searches first for the RT in the global overlay. It computes then its different identifiers as illustrated by Fig. 3, and at each derivation, it matches its new identifier with the vector of the next entity, according to what it has just learned from the RT.

NPs may also operate a control access or enable any other own mechanism (e.g., a bill service) to the nodes joining the different overlays they manage. In this case, after retrieving the RT, new nodes have to send at the global level a join intention request to the CN to know the CN's different *nodeIDs* from the CN itself.

Now, to share a resource identified by an $objectID$, the new node inserts this $objectID$ in each overlay it participates in (including the global P2P layer). The system ensures thus that the result of a query found in an entity can be retrieved through a path connection that does not pass by a node of another larger entity.

Consequently, NETPOPPS uses a key management technique to optimize data retrieval according to NP's criteria through a specific semantics.

6 Prospects

CAP [42] and NETPOPPS [43] are two different DHT-based topology aware P2P systems guaranteeing that a data item will be retrieved from the zone or entity where it has been found and the entire P2P path connexion will remain in that zone or entity. Both promise also alleviated message latency and enhanced lookup time. However, the former is more userfriendly and the latter is more network provider compliant.

In fact, CAP is a configurable and extensible context-aware system, where zones are independent from each others, and any node can create and initiate a zone according to any desired performance metric. Guarantee for context-awareness is then given by the semantic of the $objectID$ through the HKey. The system can also be adapted for semantic queries or serve as a core routing for any overlay application aiming improved security issues or quality of service. CAP is thus service oriented, and possible applications to implement over it are as numerous as values a HKey can have. However each node or object manages multiple different independent identifiers.

As for NETPOPPS, the key management is greatly simplified, as each object has only one identifier in the whole system and the identifiers of each peer are in a simple relational computation between each other. However, the system is managed by the network provider in a fixed well structured architecture where it is impossible for

users to create deliberately their own entity or group of communication, especially if they are clients of different network providers. But the control node opens wide the door to any commercial service, access control, statistics, etc. Nevertheless, NET-POPPS assume that each node knows its BE through an external procedure (it could be based on some database reports like BGP tables [2]).

Consequently, and as shown by in the oracle service [40] and P4P [41], network operators can play an important role in addressing P2P topology-awareness, traffic and costs optimization, and application performance. But operators generally consider topology of the networks they control to be confidential information [13]. Thus, in order to succeed and achieve wide adoption, any solution should provide a method to help P2P applications in peer selection without explicitly disclosing topology of the underlying network [13]. In this perspective, CAP [42], as a generic flexible userfriendly system, is promising.

However, like in any applicative research field, wide adoption will probably never happen without an agreement on a common solution based on open standards [13].

Logical routing, like the one in use in DHT-based P2P networks, will continue to evolve in the coming years, and several various new applications based on it will emerge. New claims will thus be required for corresponding architectures, e.g., security, clustering, quality of service, etc.

DHT-based structures with simple hash function are likely to be unable to afford such requirements, and existing emerging solutions (e.g., P4P [41]) are designed for dedicated infrastructures.

Our contributions aim resource lookup and retrieval within specific domains in the spirit of DHTs and with specific semantics. Large-scale evaluation of these solutions is currently in progress with the OverSim framework [46].

Future interesting research directions will then be explored to distinguish *nodeIDs* and *objectIDs*, i.e., to create one single overlay for the nodes and several overlays for the resources, each having its own semantics.

7 Conclusion

This chapter, dedicated to network-aware DHT-based P2P systems, began with elucidation of the network-aware problem. It detailed DHTs, their strengths and weakness and cleared up the incentives to focus on this specific class of P2P algorithms. The general requirements for network-awareness followed with a quick overview on available systems that explicitly provide underlay information or awareness evaluation.

Then a few well-known DHT-based topology-aware systems were briefly presented, followed by two other systems for topology estimation through layer cooperation. And as the latter allow cooperation between P2P traffic and network providers' policies, the chapter went less briefly on two more emerging architectures to translate such cooperation in semantics within the different identifiers of a DHT system. These two proposals were then compared, concluding with what could

be the trend of network-aware P2P systems, followed by a brief discussion on future research directions.

References

1. Meulle M.(2007): Interference of AS business relationships and routing policies in the Internet. PhD Thesis. Université Blaise Pascal Clermont-Ferrand, France
2. Rekhter Y., Li T.(1995): A Border Gateway Protocol 4 (BGP-4). IETF, RFC 1771
3. Karagiannis T., Rodriguez P., Papagiannaki K.(2005): Should ISPs fear Peer-Assisted Content Distribution? In: ACM SIGCOMM Proceedings of IMC. USENIX Association
4. Keralapura R., Taft N., Chuah C.N., Iannaccone G. (2004): Can ISPs Take the Heat from Overlay Networks? In: Proceedings of HotNets-III. ACM Press, New York
5. Seetharaman S., Ammar M. (2006): On the interaction between dynamic routing in the overlay and native layers. In: Proceedings of INFOCOM. IEEE Communications Society
6. Liu Y., Zhang H., Gong W., Towsley D. (2005): On the interaction between overlay routing and traffic engineering. In: Proceedings of INFOCOM. IEEE Communications Society
7. Ren S., Guo L., Zhang X. (2006): ASAP: An AS-aware peer-relay protocol for high quality VoIP. In: Proceedings of ICDCS. IEEE Computer Society
8. Rhea S. (2005): OpenDHT: A Public DHT Service. Ph.D. Thesis., University of California, Berkeley
9. Viennot L. (2005) : Pair-A-Pair: routage dans les réseaux de pair à pair. In: Panorama des Recherches Incitatives en STIC (PaRISTIC)
10. Karger D., Lehman E., Leighton F., Levine M., Lewin D., Panigrahy R. (1997): Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In: Proceedings of STOC. ACM Press, New York
11. Gummadi P.K., Gummadi R., Gribble S., Ratnasamy S., Shenker S., Stoica I. (2003): The Impact of DHT Routing Geometry on Resilience and Proximity. In: Proceedings of SIGCOMM. ACM Press, New York
12. Risson J., Moors T. (2006): Survey of Research towards Robust Peer-to-Peer Networks: Search Methods. Computer Networks 50(17):3485–3521
13. Marocco E., Gurbani V. (2008): Application-Layer Traffic Optimization (ALTO) Problem Statement. draft-marocco-alto-problem-statement-01 (work in progress)
14. Sit E., Morris R.T. (2002): Security Considerations for Peer-to-Peer Distributed Hash Tables. In: Peer-to-Peer Systems (IPTPS), number 2429 in LNCS. Springer, Berlin/Heidelberg
15. Rhea S., Geels D., Roscoe T., Kubiatowicz J. (2004): Handling Churn in a DHT. USENIX Annual Technical Conference
16. Jain S., Mahajan R., Wetherall D. (2003): A Study of the Performance Potential of DHT-based Overlays. In: Proceedings of USITS. USENIX Association
17. Li J., Stribling J., Gil T.M., Morris R., Kaashoek F.M. (2004): Comparing the performance of distributed hash tables under churn. In: Peer-to-Peer Systems-III (IPTPS), number 3279 in LNCS. Springer, Berlin, Heidelberg
18. Stoica I., Morris R., Liben-Nowell D., Karger D.R., Kaashoek M.F., Dabek F., Balakrishnan H. (2001): Chord: A Scalable Peer-to-peer Lookup Protocol for Internet Applications. In: Proceedings of SIGCOMM. ACM Press, New York
19. Rowstron A., Druschel P. (2001): Pastry: Scalable, decentralized object location and routing for large-scale peer-to-peer systems. In: IFIP/ACM Middleware 2001, number 2218 in LNCS. Springer, Berlin/Heidelberg
20. Ratnasamy S.: (2002) A Scalable Content-Addressable Network. Ph.D. Thesis, University of California at Berkeley
21. Zhao B.Y., Huang L., Stribling J., Rhea S.C., Joseph A.D., Kubiatowicz J.D. (2004): Tapestry: A Resilient Global-Scale Overlay for Service Deployment. IEEE J-SAC 22(1):41–53

22. Maymounkov P., Mazières D. (2002): Kademlia: A peer-to-peer information system based on the XOR metric. In: Peer-to-Peer Systems (IPTPS), number 2429 in LNCS. Springer, Berlin/Heidelberg
23. Malkhi D., Naor M., Ratajczak D. (2002): Viceroy: A Scalable and Dynamic Emulation of the Butterfly. In: Proceedings of PODC. ACM Press, New York
24. Fraigniaud P., Gauron P. (2006): D2B: a de Bruijn Based Content-Addressable Network. Theoretical Computer Science 355(1):65–79
25. Loewenstern A. (2008): DHT Protocol. BitTorrent.org `http://www.bittorrent.org/beps/bep0005.html. Accessed20August2008`
26. John (2004): eMule v.40f is available. eMule-Project.net `http://www.emule-project.net.Accessed20August2008`
27. Xu Z., Tang C., Zhang Z.(2002): Building Topology-Aware Overlays using Global Soft-State. HPL-2002-281. Hewlett-Packard Laboratories, Palo Alto
28. Ratnasamy S., Handley M., Karp R., Shenker S. (2002): Topologically aware overlay construction and server selection. In: Proceedings of INFOCOM. IEEE Communications Society
29. Cox R., Dabek F., Kaashoek F., Li J., Morris R. (2003): Practical, distributed network coordinates. In: Proceedings of HotNets-II. ACM Press, New York
30. Madhyastha H.V., Isdal T., Piatek M., Dixon C., Anderson T., Krishnamurthy A., Venkataramani A. (2006): iPlane: an information plane for distributed services. In: Proceedings of OSDI. USENIX Association
31. Castro R., Coates M.J., Liang G., Nowak R., Yu B. (2004): Network Tomography: Recent Developments. Statistical Science 19(3):499–517
32. Rostami H., Habibi J. (2007): Topology awareness of overlay P2P networks. Concurrency and Computation: Practice & Experience 19(7):999–1021
33. Castro M., Druschel P., Hu Y.C., Rowstron A. (2002): Topology-Aware Routing in Structured Peer-to-Peer Overlay Networks. MSR-TR-2002-82. Microsoft Research, Redmond
34. Ratnasamy S., Shenker S., Stoica I. (2002): Routing Algorithms for DHTs: Some Open Questions. In: Peer-to-Peer Systems (IPTPS), number 2429 in LNCS. Springer, Berlin/Heidelberg
35. Joseph A.D., Zhao B.Y., Duan Y., Huang L., Kubiatowicz J.D. (2002): Brocade: Landmark Routing on Overlay Networks. In: Peer-to-Peer Systems (IPTPS), number 2429 in LNCS. Springer, Berlin/Heidelberg
36. Xu Z., Mahalingam M., Karlsson M. (2003): Turning Heterogeneity into an Advantage in Overlay Routing. In: Proceedings of INFOCOM. IEEE Communications Society
37. Xu Z., Min R., Hu Y. (2003): Hieras: A DHT Based Hierarchical P2P Routing Algorithm. In: Proceedings of ICPP. IEEE Computer Society
38. Garcés-Erice L., Ross K.W., Biersack E., Felber P.A., Urvoy-Keller G. (2003): TOPLUS: Topology Centric Lookup Service. In: Group Communications and Charges (NGC), number 2816 in LNCS. Springer, Berlin/Heidelberg
39. Ferreira R.A., Grama A., Jagannathan S. (2004): Plethora: An Efficient Wide-Area Storage System. In: Bougé L., Prasanna VK (eds) High Performance Computing (HiPC), number 3296 in LNCS. Springer, Berlin/Heidelberg
40. Aggrawal V., Feldmann A., Scheideler C. (2007): Can ISPs and P2P Users Cooperate for Improved Performance. In: ACM SIGCOMM Computer Communications Review 37(3): 31–40
41. Xie H., Krishnamurthy A., Silberschatz A., Yang Y.R. (2007): P4P: Explicit Communications for Cooperative Control Between P2P and Network Providers. P4PWG Whitepaper
42. Fayçal M., Serhrouchni A. (2007): CAP: A Context-Aware Peer-to-Peer System. In: Meersman R., Tari Z., Herrero P (eds) On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops, number 4806 in LNCS. Springer, Berlin/Heidelberg
43. Fayçal M., Serhrouchni A. (2008): NETPOPPS: A Network Provider Oriented Peer-to-Peer System. In: IFIP Proceedings of NTMS. IEEE Communications Society

44. Krawczyk H., Bellare M., Canetti R. (1997): HMAC: Keyed-Hashing for Message Authentication. IETF, RFC 2104
45. Hassan H.R., Bouabdallah A., Bettahar H., Challal Y. (2005): An Efficient Key Management Algorithm for Hierarchical Group Communication. In: IEEE, CreateNet Proceedings of SecureComm. IEEE Computer Society
46. Baumgart I., Heep B., Krause S. (2007): OverSim: A Flexible Overlay Network Simulation Framework. In: Proceedings of Global Internet Symposium (GI) in conjunction with Infocom 2007. IEEE Communications Society