Context-Aware Location in the Internet of Things

Federico Andreini, Flavio Crisciani, Claudio Cicconetti, Raffaella Mambrini
Intecs S.p.a.
Via Egidio Giannessi, 5 – I-56126 Pisa, Italy

***Abstract*—The Internet of Things is rapidly diffusing, though severe modifications are needed to the current Internet architecture to cope with its new requirements. A major step forward is decoupling the name of an object (identifier) from its physical location (locator), which are both expressed by the IP address currently. We argue that additional advantages can be brought if context metadata are provided along with the locator, thus forming an *enriched locator*. In this work, we propose an efficient architecture that allows the enriched locators to be distributed efficiently and in a scalable manner. The distribution protocol exploits the multiple Distributed Hash Table concept, which is well established in peer-to-peer overlays. Preliminary evaluation results obtained with a practical testbed implemented are reported.**

I. Introduction

The need for a new architecture of the Internet is generally accepted [1], due to the constantly increasing number of limitations going under the spotlight whenever a new technology, service, or application is put forward. One big opportunity that is being currently investigated in academy and industry alike is the introduction of a Service-Oriented Architecture (SOA), allowing physical objects to publish services that are accessed by mobile clients to enable flexible human-to-machine and machine-to-machine (M2M) interactions [2]. This translates into a paradigm shift from host-centric to content-centric, as discussed, e.g., in [3], where the authors also analyze the benefits of providing an ISP with the means to offer a native support to typical data dissemination service without using overlays. A structural change that is required to enable this shift is the decoupling of the two roles currently played by the IP address [4], i.e. identifier and locator. As a matter of fact, this overload also creates problems due to the routing table size growth, and additional concerns for supporting mobility and security, as mentioned in [5].

One of the first proposals to implement the distinction between identity and location has been proposed in [6], where the authors describe a routing system based on a label representing the host identity. Later on the distinction between resources and hosts in which they are stored has been emphasized in [7], where the authors discuss some design considerations of a global networking architecture defined within the 4WARD EU project[1], called NetInf. The Content Centric Network (CCN) presented in [8] is a different implementation of a global networking architecture, which can be deployed at almost any level of the standard TCP/IP stack. In fact, it relies on standard routing protocols, which however forward/route data based on structured resource names rather than on IP addresses. In [9] the authors propose a routing architecture based on the User Identifier Protocol (UIP), where the use of hierarchical tables is proposed as a solution for the locator lookup problem, in a sort of distributed and identifier-based DNS.

In our view, the advantages of a clear separation between identifier and locator can be pushed further if the latter is accompanied by some context metadata, which is any kind of information that might be useful to the service subscriber to better exploit the service, thus becoming an *enriched locator* (e-locator). In fact, the benefits of context-awareness are being currently investigated and exploited for a variety of networks and services, e.g. [10]. An example of context provided by the metadata fields of the e-locator is reported for illustration purposes for the application of video surveillance and monitoring. Furthermore, we propose an architecture for distributing the e-locators. The proposed distribution protocol is based on multiple Distributed Hash Table (DHT) routing and it has been designed to be especially beneficial for Web of things applications, like M2M communication, involving embedded devices. We developed a practical implementation of the distribution network and set up a small-scale testbed. The experimental results obtained show that the proposed approach is promising in terms of scalability and efficiency.

The rest of this document is structured as follows. The proposed innovations are described in details in Section II and are assessed in Section III. Conclusions are drawn in Section IV.

II. Proposed Architecture

In this section we discuss the innovative concept of e-locator and describe a possible implementation inspired from peer-to-peer (P2P) systems to distribute them efficiently. In a SOA, objects of the physical world, possibly mobile, are interconnected and provide the users, or subscribers, in the network with a portfolio of services. In order to use a given service, the subscriber needs to know the unique identifier (ID) of the service and the physical or logical location (locator) to reach the provider. As already mentioned, currently these two functions coexist within the IP address, which is inefficient for a variety of reasons (e.g., see [11]). In this work we focus on the location problem only. We assume that there is a means to obtain the ID of a service, i.e. a sequence of bits, from a high-level name, i.e. some human-friendly string of characters like an Internet Uniform Research Identifier (URI) or a DOI.

In our proposed architecture, a provider publishes a service by distributing an e-locator = {locator + metadata}. The

[1] http://www.4ward-project.eu/

978-1-4244-8865-0/10/$26.00 ©2010 IEEE

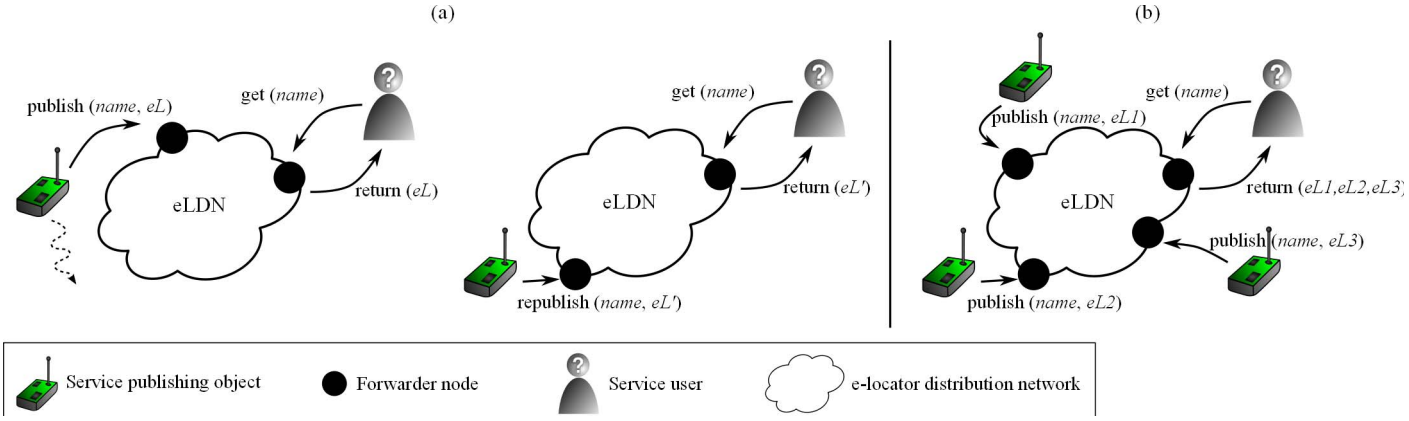

Fig. 1. High-level view of the functions of the e-locator Distribution Network (eLDN) with (a) mobility and (b) replication.

locator allows a network-level end-to-end connection between the publisher and the subscriber, while the metadata fields add *context* to the locator. We call the logical infrastructure to publish the e-locators *e-locator Distribution Network* (eLDN), which is described later on in Section II-A. We assume for simplicity that the locator is an IPv4 or IPv6 address, which allows our proposed architecture to be implementable in today's Internet.

The metadata fields can contain pieces of information useful at either the application or the layers below for a more efficient reachability or exploitation of the service provided. Examples of such fields can be the geographical location of the object, the technology used for communication, or security data. To enable an incremental evolution of the applications, services, and technologies supported, we propose to use the Extensible Markup Language (XML) to transport the metadata fields of the e-locator. An example of set of metadata fields is reported below in Section II-C. Instead, we focus now on the definition of an efficient architecture for the eLDN, whose functional requirements are sketched in the following.

A high-level view is provided in Fig. 1, which illustrates two use cases of particular interest for emerging applications. In the left-hand there is a physical object publishing a service by means of an e-locator eL, associated to a given service addressed by its $name$, which will be translated into an ID by means of some resolution system. The basic function of the eLDN is to return eL to any user requesting the service via $name$, as illustrated in the picture, by coping with possible *mobility* of both the physical objects and the users. Another important feature that must be supported by an eLDN is *replication*, which is illustrated in the right-hand side of Fig. 1. In fact, it can happen that the very same service is provided by more than one object. Therefore, the eLDN must provide the user with a (possibly partial) list of the e-locators. The user will then be able to choose the best one to contact based on the metadata fields in a context-aware manner.

A. eLDN Architecture

To support the features discussed above, we propose to employ an architecture based on unstructured P2P: the use of a P2P-based architecture in mobile environments has been endorsed in [12] and replication is easily supported. Additionally, P2P architectures have been successfully demonstrated to be both scalable and reliable (see, e.g., [13]), as well as resilient to flash-crowd effects and denial-of-service (DoS) attacks due to their fully distributed nature.

The basic concept of unstructured P2P is the DHT. A traditional hash table in programming languages is made of pairs $(key, value)$, where every $value$ can be retrieved from its key at some cost depending on the implementation. A *distributed* hash table is similar in principle, but the data are not stored in a single physical location in memory but, rather, are distributed as uniformly as possible among a set of nodes, called *peers*. In our eLDN the key is a hash of the name, and the value is the e-locator. The peers are all the nodes acting as the network access points for both the publishers and the subscribers, and they are also assigned a key (like services) for simplicity of implementation. We envisage that both the publishers (e.g., sensors) and the subscribers (e.g., mobile phones) might be subject to technology constraints, in terms of the computational and communication capabilities, stemming from the their mobile nature. Therefore, we assume that the eLDN is maintained only by those nodes part of the (mostly wireless, but static) infrastructure, like access points, base stations, and routers, whose number and computational capabilities are growing at a constant pace to make the dream of ubiquitous networking a reality. Furthermore, this two-tier logical structure allows a decoupling that is beneficial when the publisher or the subscriber are confined within the access network for security or administrative reasons.

The publish and retrieval phases[2] are illustrated in Fig. 2. The left-hand side of the figure shows a service provider that connects to a peer of the eLDN, called its *anchor node*, to advertise an e-locator associated to a given name. The anchor node obtains the key by applying the hash function to the name. The key is then used in an iterative process to find the so-called *home node*, i.e. the peer which is responsible

[2]Due to limit page budget, we do not address all the details here, e.g. for the procedures for a node to join or leave the network.

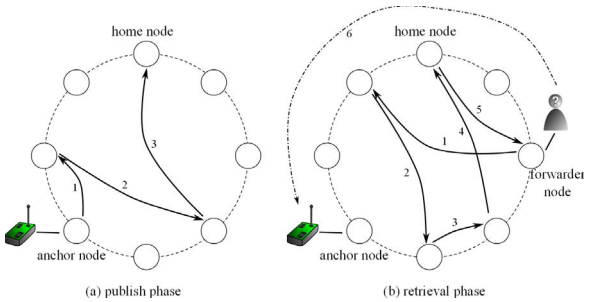

Fig. 2. DHT-based implementation of the eLDN.

for actually storing the e-locator. Since all the peers use the same lookup process, every subsequent request of the same key started from a so-called *forwarder node* will eventually reach the home node, which can then reply back directly to it as depicted in the right-hand side of Fig. 2. This way, replication is automatically supported, since all the services with the same name/key will be directed to the same home node, which keeps a table where a list of e-locators is mapped to each key. As a further optimization, during the publishing process the intermediate nodes, including the anchor node itself, may opportunistically cache the pairs so as to enable an early termination of the retrieval phase if a request for the cached name passes through them. This would, however, require a mechanism for handling the inconsistency if a service is removed from the eLDN, which is left for future work.

We note that the e-locators are re-distributed among the peers so as to achieve a uniform dissemination in the network, in accordance with the basic properties of the DHT, which have been demonstrated both in theory and in practice (e.g., [14]). Another desirable property is that the number of hops between the anchor/forwarder node and the home node has a theoretic upper bound equal to $log_{2^B}(N)$ [15], where B is the number of bits used to represent a digit of the key and N is the number of peers. This explains the excellent scalability properties of P2P systems, where the number of hops, which impacts on both the amount of network traffic and the retrieval/publishing latency, has a logarithmic increase with the number of nodes, whereas the storing requirements per node *decrease* linearly with the number of nodes. Even though not discussed in details here due to page limitations, additional hierarchical levels can be added, thus creating a multiple DHT, to achieve a further degree of scalability on a geographical scale

B. Implementation

A proof-of-concept implementation of the eLDN has been developed for demonstration purposes. Advanced optimizations, e.g., topology awareness [16], have not been considered so far but can be included without impacting the overall framework. To combine efficiency and reusability we used the C++ programming language, along with the Boost libraries[3] for socket and multi-thread programming, and the Google Protocol Buffers[4] for structured data encoding. For simplicity TCP is used as the transport layer protocol both for the connection between the publisher (subscriber) and its forwarder (anchor) node and for the connections of peers. A more lightweight implementation using UDP, to remove the overhead of establishing and tearing down TCP connections, is feasible with minor modifications.

Additionally, we have developed an Application Programming Interface (API) in both C++ and Java that allows a developer to connect her application to the forwarder/anchor node and, hence, create a service provider or user. The following basic interface functions are defined:

- `boolean publish (String name, String eL)`
- `boolean unpublish (String name, String eL)`
- `String[] get (String name)`

where the first two functions are used by the objects to publish and remove, respectively, a service and the last one is used by the users to retrieve the list of e-locators corresponding to a given service name.

Every peer keeps the following data structures for the routing and e-locator distribution functions. The *metatadata table* is queried by the home nodes in order to retrieve the metadata associated to a given key: indeed it is implemented as a map using the ID as key and a list of e-locators as value. Two thresholds are configured to limit the amount of e-locators contained within this table, total and per-ID, which avoid undesirable memory overrun situations. The pending requests for a given ID are stored in the *request table*: when a reply message containing the e-locator(s) arrives at the forwarder node matching an entry of the table, it is delivered to all the requesting subscribers. Like for the metadata table, there are two thresholds intended to control the number of overall and per-ID requests.

Additionally, every peer is provided with a set of three tables providing P2P forwarding and routing capabilities in a similar manner as Pastry [15]: the *leaf set* stores the <ID,locator> pairs of the logical neighbors, in terms of numerical ID proximity; the *neighbouring table* stores the <ID,locator> pairs of the physical neighbors and it is used to minimize the number of forwarding hops; finally, the *routing table* collects the pairs of the nodes acquired during the join procedures which are not contained in the leaf set. All the tables are currently implemented using the STL map container[5], which yields a logarithmic time lookup. Further development may consider more efficient containers as the Google Sparse Hash[6] or other structures providing a pseudo-constant element access-time without compromising the implementation framework.

Finally, the ID is implemented as a 20-byte stream obtained from a SHA1 hash function, which can be easily changed if necessary, e.g., to support a higher number of nodes or resources.

[3]http://www.boost.org/

[4]http://code.google.com/p/protobuf/

[5]http://www.sgi.com/tech/stl/

[6]http://code.google.com/p/google-sparsehash/

```
<locator>
  <ipv4>1.2.3.4</ipv4>
  <protocol>tcp</protocol>
  <port>1234</port>
</locator>
<metadata>
  <gps>43716164,10396492</gps>
  <video>
    <codec>mpeg2</codec>
    <fps>18</fps>
    <resolution>640,480</resolution>
  </video>
  <security>
    <accessType>open</accessType>
  </security>
</metadata>
```

Fig. 3. Example of an e-locator, encoded in XML.

C. E-locator example

We now report an example of usage of the e-locator with a target application to show that it is useful for practical uses. The quantitative evaluation of its benefits, compared to using the mere locator, is left for future work. The application selected is video surveillance, which can serve different purposes, from public security to virtual sightseeing for tourists, from live weather feedbacks to active monitoring of restricted areas. In this scenario the service providers are cameras installed in fixed locations, and the subscribers are the different actors that need to establish a network layer connection with a camera, e.g., public officers, tourists. The camera is connected to a wired or wireless router, which we suppose acts as its anchor node.

An example of e-locator for this application is reported in Fig. 3. The locator consists of the IPv4, the transport protocol type, and the port that should be used to connect to the camera. Note that, because of the flexibility provided by this structure, adding or substituting an IPv6 locator, or providing multiple alternatives for the transport protocol, would be straightforward. In addition to the connectivity alone, we identified the following metadata fields that provide the subscriber's application with enriched context information: a Global Positioning System (GPS) tag, which gives the absolute geographic position of the camera; video tags including, e.g., codec, frame rate, and resolution of the video flow that can be provided by the camera; finally, security tags, which carries information about the security capabilities of the camera, such as accounting or the supported cryptographic algorithms (in the example the camera access is open to all). The GPS and video-specific tags can be used in those cases when the resolution process returns more than one e-locator for the given service name/identifier so that the subscriber's application can choose the most suitable alternative based on the device capabilities and user-configured preferences. Security issues are not tackled directly by the proposed framework itself, but the context data included in the e-locator can give early indications to the users on whether permission to use the camera data can be granted or not, without the need to connect to the camera and, possibly, have the service refused.

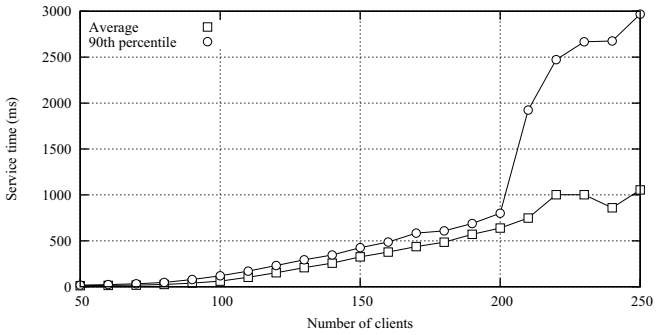

Fig. 4. Service time vs. clients, with a fixed amount of resources.

III. Evaluation

In this section we report the experimental results obtained with a practical implementation of the proposed solution.

An eLDN of 41 peers has been set up for demonstration purposes: one peer resides in an embedded PC[7] with limited computational capabilities connected via a LAN to a fully-capable workstation emulating the other 40 peers. A variable number of clients perform requests to the eLDN passing through the embedded PC, which acts as the forwarder node for all the clients and is a natural performance bottleneck. Each request is made for an identifier that is drawn from the set of published ones. All the clients are emulated into a dedicated workstation. After each request, the client waits for a random amount of time, drawn from an exponential distribution with average 500 ms, before performing the next request. Service time, defined as the time between when the client passes a request to the forwarder node and the time when the client receives a response from the forwarder node, has been considered as a performance metric while varying the network load. The evaluation of scenarios with a higher number of nodes geographically distributed, by means of either a global research network (e.g. PlanetLab) or network simulation, are left for future work.

In the first set of experiments, we increase the number of clients from 50 to 250, with a fixed number of resources published, equal to 150. In Fig. 4 we report the average service time and its 90th percentile. As can be seen, both curves increase slowly until 200 clients. Such an increase is due to the higher chance that multiple requests are enqueued at the forwarder node at the same time, in which case an incoming request has to wait until all the pending requests are dispatched. Note that dispatching a request incurs a time overhead due to the TCP three-way handshake and the creation of a new serving thread. After 200 clients, the 90th percentile has a sharp increase, due to embedded PC not being able to satisfy the incoming requests fast enough. The average curve does not show the same behavior as the 90th percentile,

[7]ALIX board alix6e2 (http://www.pcengines.ch/alix6e2.htm), mounting a 500 MHz AMD Geode LX800 CPU and 256 MB of RAM.

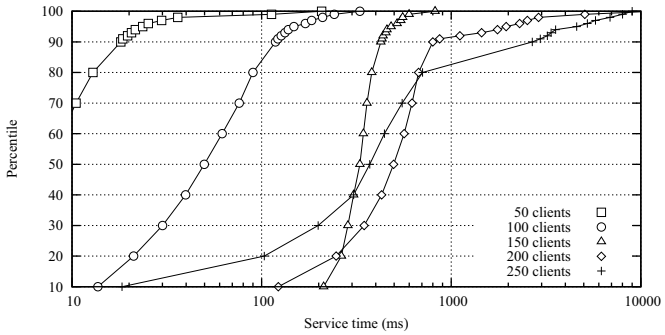

Fig. 5. Percentiles of service time vs. clients, with a fixed amount of resources.

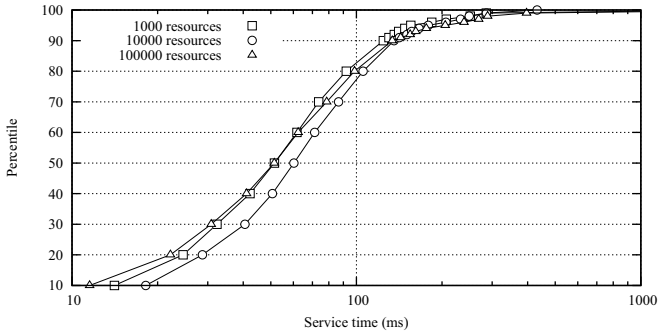

Fig. 6. Percentiles of service time vs. resources, with a fixed amount of clients.

because of a caching phenomenon, which is explained in details next.

In Fig. 5 we show the percentiles of the service time obtained in a subset of the experiments above. As far as the high percentiles, i.e., above 80th, are concerned, the curves move to the right (i.e. the latencies increase) as the number of clients increases. This is because the probability of having long queues of pending requests increases. However, there is a compensating effect on the lower percentiles for more than 100 clients, because the forwarder node keeps track of the requests that are pending for the same identifier and, after performing a single P2P query, will respond to all the requesting subscribers in a batch. Such an effect is more prominent when the size of the queue of pending requests is larger: with a fixed amount of resources and an increasing number of requests per second such an occurrence increases, as well. This explains why the curves with 150, 200, and 250 clients intersect at some points, while the average curve in Fig. 4 does not increase as dramatically as the 90th percentile.

We conclude by reporting the results from a second set of experiments, where we varied the number of resources published from 1,000 to 100,000, while setting to 100 the number of clients. In the current implementation data structure search time is logarithmic with respect to the number of resources. In Fig. 6 we report the percentiles of the service time. As can be seen, the results are only slightly affected by the increasing number of resources published. Indeed, the curves are almost overlapping, showing that the search time within the internal data structures of the P2P overlay does not create a performance bottleneck notwithstanding the limited computational capabilities of the embedded PC.

IV. Conclusion

In this paper we have proposed the concept of e-locator, which allows context data to be distributed together with the locator in a SOA for a more efficient exploitation in the Internet of Things, especially for M2M applications. As an example, we have described how to use the e-locator to distribute pieces of information useful for the specific application of video surveillance and monitoring. Furthermore, we have defined a practical implementation of a distribution network of the e-locators, based on a P2P paradigm. The proposed solution has been implemented and evaluated in a testbed. The results of the experiments show that high scalability can be achieved in practice, even with hardware having limited computational capabilities, which are commonly found in the access segment of today's wireless networks.

References

[1] Future Content Networks Group, "Why do we need a Content-Centric Future Internet ?" 2009.

[2] D. Guinard and V. Trifa, "Towards the Web of Things : Web Mashups for Embedded Devices," in *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM)*, 2009.

[3] M. D. Ambrosio, P. Fasano, M. Marchisio, V. Vercellone, and M. Ullio, "Providing Data Dissemination Services in the Future Internet," in *IEEE Globecom*, 2008.

[4] I. S. Shenker, "A Data-Oriented (and Beyond) Network Architecture," in *ACM SIGCOMM*, 2007.

[5] D. Meyer, L. Zhang, and K. Fall, "RFC4984: Report from the IAB Workshop on Routing and Addressing," 2007.

[6] M. Caesar, T. Condie, J. Kannan, K. Lakshminarayanan, and I. Stoica, "ROFL: Routing on Flat Labels," *ACM SIGCOMM Computer Communication Review*, vol. 36, no. 4, p. 363, Aug. 2006.

[7] B. Ahlgren, V. Vercellone, M. D'Ambrosio, M. Marchisio, I. Marsh, C. Dannewitz, B. Ohlman, K. Pentikousis, O. Strandberg, and R. Rembarz, "Design considerations for a network of information," in *ACM CoNEXT*. New York, New York, USA: ACM Press, 2008.

[8] V. Jacobson, D. K. Smetters, J. D. Thornton, M. F. Plass, and N. H. Briggs, "Networking Named Content," *ACM CoNEXT*, p. 1, 2009.

[9] H. Li, P. Chenghui, and B. Li, "User ID Routing Architecture," *IEEE Vehicular Technology Magazine*, 2010.

[10] J. Simoes and T. Magedanz, "Contextualized User-Centric Multimedia Delivery System for Next Generation Networks," *Telecommunication Systems*.

[11] H. Li, C. Peng, B. Li, Y. Chen, W. Zhang, J. Wu, and H. Huang, "A Sustainable Wireless Internet with Simplicity, Efficiency, and Secured User Sharing," *IEEE Vehicular Technology Magazine*, vol. 3, no. March 2010, pp. 62–69, 2009.

[12] S. Pack, K. Park, T. Kwon, and Y. Choi, "SAMP: Scalable Application-Layer Mobility Protocol," *IEEE Communications Magazine*, no. June, pp. 86–92, 2006.

[13] S. Ioannidis and P. Marbach, "Absence of Evidence as Evidence of Absence: A Simple Mechanism for Scalable P2P Search," in *IEEE INFOCOM*. Ieee, Apr. 2009, pp. 576–584.

[14] X. Shen, H. Yu, J. Buford, and M. Akon, Eds., *Handbook of Peer-to-Peer Networking*. Springer, 2010.

[15] D. Rowstron, "Pastry Scalable, decentralized object location and routing for large-scale peer-to-peer systems," in *IFIP/ACM International Conference on Distributed Systems Platforms (Middleware)*, 2001.

[16] S. Ren, E. Tan, T. Luo, S. Chen, L. Guo, and X. Zhang, "TopBT : A Topology-Aware and Infrastructure-Independent BitTorrent Client," in *IEEE INFOCOM*, 2010.