SPAD: A distributed middleware architecture for QoS enhanced alternate path discovery

Thierry Rakotoarivelo [a,b,c,*], Patrick Sénac [b,d], Aruna Seneviratne [c], Michel Diaz [d]

[a] *School of Electrical Engineering and Telecommunications, University of New South Wales, Sydney, NSW 2052, Australia*
[b] *Department of Applied Mathematics and Computer Engineering, ENSICA, Place Emile Blouin, 31056 Toulouse, France*
[c] *National ICT Australia, Locked Bag 9013, Alexandria, NSW 1435, Australia*
[d] *LAAS-CNRS, 7 avenue du Colonel Roche, 31077 Toulouse, France*

Abstract

In the next generation Internet, the network will evolve from a plain communication medium into one that provides endless services to the users. These services will be composed of multiple cooperative distributed application elements. We name these services *overlay applications*. The cooperative application elements within an *overlay application* will build a dynamic communication mesh, namely an *overlay association*. The Quality of Service (QoS) perceived by the users of an *overlay application* greatly depends on the QoS experienced on the communication paths of the corresponding *overlay association*. In this paper, we present super-peer alternate path discovery (SPAD), a distributed middleware architecture that aims at providing enhanced QoS between end-points within an *overlay association*. To achieve this goal, SPAD provides a complete scheme to discover and utilize composite alternate end-to-end paths with better QoS than the path given by the default IP routing mechanisms.

Keywords: Quality of Service; Overlay networks; Peer-to-peer; Service-oriented networks; Middleware

1. Introduction

In recent years, the presence of mobile network-enabled devices in our environment has steadily increased. These devices will collectively form a pervasive computing and networking environment around the users, informing them, satisfying their communication needs, and performing various tasks on their behalf. In that regard, it might be unrealistic to assume that these portable devices will be capable of providing the increasing and extensive local applications, computing or storage resources that the users will require. In contrast, it might be more realistic to rely on service providers at the edge of the network to provide the required resources in a

* Corresponding author. Address: National ICT Australia, Locked Bag 9013, Alexandria, NSW 1435, Australia. Tel.: +61 2 8374 5245; fax: +61 2 8374 5531.
E-mail addresses: thierry@mobqos.ee.unsw.edu.au (T. Rakotoarivelo), senac@ensica.fr (P. Sénac), Aruna.Seneviratne@nicta.com.au (A. Seneviratne), Michel.Diaz@laas.fr (M. Diaz).

distributed manner. In this approach, the end-systems will view the network as an endless source of services, and will act mainly as input/output devices. This approach will benefit the mobile devices in terms of cost, mobility support and lower energy consumption. As discussed in [1], virtual test-beds will facilitate the initial deployment of such service-oriented networks. These service-oriented networks will provide composite services to the users. These services will be composed of multiple cooperative distributed software elements, as described in [2]. These software elements will be elementary services, which will perform generic tasks and communicate with each other, via overlay networks that will be dynamically created over the existing Internet infrastructure as required. We use the term *overlay application* to describe a distributed application composed by such distributed elementary services. Fig. 1 illustrates the deployment of an *overlay application S* between two hosts, *X* and *Y*. *S* is composed of the software elements (i.e. elementary services) *S*1, *S*2, *S*3, and *S*4, respectively provided by hosts (i.e service servers) *A*, *B*, *C* and *D*. This figure also describes an example of a typical *overlay application*. In this example, *S* is a complex video-conferencing application, which performs tasks such as video-adaptation or subtitle generations (implemented by the independent software elements S_is).

Within an *overlay application*, data flows no longer travel between two end-points. They may instead traverse multiple peer end-points (hosting elementary services) and be produced/consumed by several other peers. For example, an audio flow in a distributed Voice Over-IP service such as Skype [3] might pass through several peers providing elementary services such as firewall traversal, or codec/quality adaptation. This defines a new peer-to-peer communication paradigm, involving a set of connections between elementary services within an *overlay application*, and complementing the traditional point-to-point, or point-to-multipoint communication paradigms. We use the term *overlay association* to refer to such a set of connections [4], as illustrated in Fig. 1. This concept of *overlay association* allows the design of mechanisms that improve and manage the Quality of Service (QoS) experienced by the users of an *overlay application* as a whole (e.g. reducing latency in the above VoIP service). These mechanisms would perform global resource optimization on an *overlay association*, rather than less efficient local resource management decisions. More precisely, these mechanisms would be part of a middleware framework [4] that would provide a unified means of managing the communication needs of any *overlay application*, thus reducing their implementation complexity. This middleware framework would mediate between the software elements involved in an *overlay application* and the network.

Super-peer based alternate path discovery (SPAD) is a distributed peer-to-peer scheme that enhances the QoS between two peers of an *overlay association*. It is a key part of the aforementioned middleware framework for overlay networks. The fundamental idea behind SPAD is to discover and utilize alternate composite Internet paths [5] that provide better QoS than the default path given by the IP routing mechanisms, similar to the approach discussed in [6]. We use the term QoS Enhanced Alternate Paths (QEAPs) to refer to such alternate paths. Fig. 2 illustrates the deployment of a QEAP between two hosts, *A* and *B*, via a third relay host *R*. More details on QEAPs and their characteristics are given in appendix A, and in an earlier contribu-

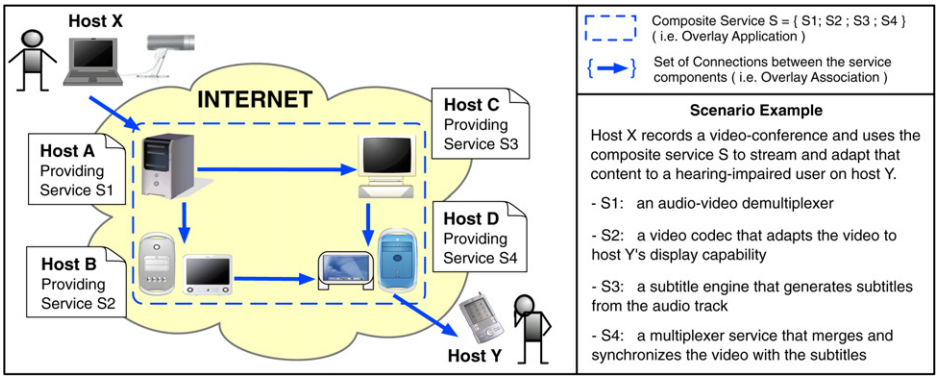

Fig. 1. Deployment of an overlay application *S* (composed of services *S*1, *S*2, *S*3 and *S*4) between hosts *X* and *Y*.

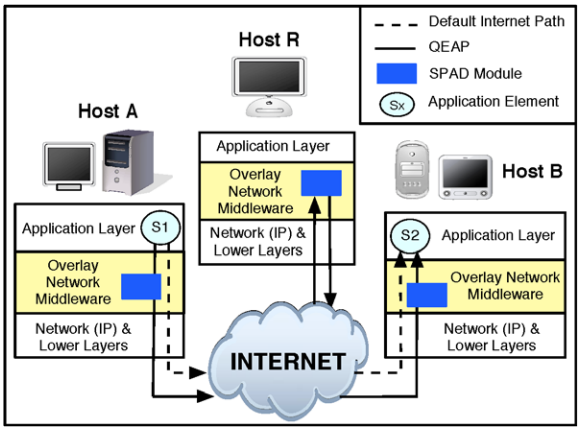

Fig. 2. Comparison between a QEAP (from host *A* to *B*, via relay host *R*) and the default Internet path.

tion on QEAPs analysis [4]. The current SPAD system allows the discovery of 2-hop QEAPs that provide better, i.e. shorter, delay between end-points. As shown in appendix A, QEAPs with more than 2 hops do not provide significant extra benefits. Furthermore, this SPAD system could be adapted to discover QEAPs that enhance other QoS parameters (e.g. bandwidth, loss). Finally, using algebraic properties of QoS parameters (e.g. delays being additive) [7], in conjunction with SPAD, it is possible to provide enhanced QoS to an entire overlay association.

This paper presents the design and experimental evaluation of SPAD. It is a complete QEAP discovery system based on a community of cooperative end-points grouped as an unstructured Super-Peer network [8]. Within this community, some end-points act as normal-peers and originate requests for information on desired alternate paths to some destinations. Other end-points act as super-peers and process these requests to determine alternate paths. They forward the requests to other super-peer neighbors, exchange connectivity information related to the SPAD community, and return the relevant information back to the requesting peers. We evaluated SPAD in different simulation environments using real-world one-way and RTT delay measurements. In these experimental environments, SPAD managed to discover significantly more than half of all existing QEAPs.

SPAD makes the following original contribution: it provides a uniform framework for the discovery of alternate paths between end-points in an efficient, transparent, and distributed manner. This enables the provision of potentially enhanced QoS for communications between these end-points. SPAD empowers the end-users at the edge of the network, allowing them to directly discover and utilize QEAPs, without having to rely on any central entity or third-parties at the Internet Service Provider (ISP) level. It is an incrementally deployable, scalable system with no single point of failure.

The remainder of this paper is organized as follows. In Section 2, we discuss some related work. Section 3 presents an overview and a design description of the SPAD architecture. Section 4 describes in details the SPAD QEAP discovery functions and mechanisms. In Section 5, we present and discuss the experimental performance evaluation of SPAD. Finally, Section 6 concludes this paper and provides some directions on our future work.

2. Related work

Savage et al. [6] discussed the existence and benefits (enhanced or controlled QoS and robustness) of alternate Internet paths. They found that QEAPs exist for up to 80% of the node pairs in their North American dataset. Following contributions such as RON [9], Q-RON [10], and ALSW [11] proposed methods and architectures to discover and utilize such paths. All these frameworks involve either a central entity (RON, ALSW) or several distributed third-party brokers (Q-RON) that sell QEAPs to end-users. Our approach differs from their works, as SPAD is a fully distributed scheme that allows a community of users to directly discover and utilize QEAPs.

Within a distributed QEAP discovery scheme, nodes play the role of both client and server, hence the choice of a Peer-to-Peer (P2P) approach in SPAD's design. The P2P paradigm has become the focus of several research studies and the basis for many popular applications such as content search and distribution, as in Gnutella [12], or real-time communication, as in Skype [3]. In [13], the authors proposed a classification of P2P systems into two models, namely structured and unstructured. SPADs architecture is based on the unstructured model. In [14], we presented an alternative basic QEAP discovery scheme based on the structured models. To the best of our knowledge, only Fei et al. [15] proposed another distributed scheme to select and utilize alternate paths on P2P overlay networks. However, their approach is aimed at backup alternate paths and not QoS enhanced ones.

Furthermore, they provided an heuristic to select a best backup alternate path among a set of potential ones, but they did not address the issue of discovering this initial set of candidate alternate paths. In contrast, SPAD allows the discovery of such an initial set, while trying to ensure that it contains QoS enhanced paths. Indeed, in some cases (such as a P2P overlay with few participating nodes) there might not exist any QEAPs. Finally, with minor modifications, SPAD could also be used to discover backup alternate paths as in [15].

As will be described in Section 4.3, SPAD features a background information dissemination algorithm among its super-peer nodes. It has been shown that percolation-based techniques, such as probabilistic flooding [16], provide efficient information dissemination with lower bandwidth usage than the classic flooding technique, and without any membership management as in Gossip-based techniques [17]. Furthermore, in [18] the authors demonstrated that probabilistic flooding in power-law unstructured P2P networks is effective in terms of bandwidth consumption. Section 4.3 extends the analysis from [18] to integrate probabilistic flooding in SPAD proactive information exchange scheme. Percolation-based random-walk also has low bandwidth requirements, but it achieves full information dissemination at a higher latency than probabilistic flooding, thus making it unsuitable for SPAD.

In an earlier contribution [4], we proposed a simple P2P QEAP discovery scheme based on a 2-level flooding algorithm. This initial scheme assumed an unstructured community of cooperative peers in which all participants are willing to share a fixed amount of their resources, to assist in discovering QEAPs and relaying traffic. Each peer N_X maintains a fixed-size list of other "close" peers, in terms of delay. These "close" peers are potential relay nodes for N_X. N_X first searches among them for a QEAP toward N_Y. If none are found, then N_X searches among the peers in N_Y's list. We evaluated this scheme using trace-based simulations on delay measurement datasets from the RIPE-TTM project [19]. We showed that given some initialization parameters, this initial scheme allowed the discovery of about 77.6% of existing QEAPs among the nodes in these datasets. Moreover due to the 2-level design, in about 86.5% of the successful discovery cases, the incurred message cost among the peers was equal to $2 * \log N$ (N being the number of peer in the community). This cost was equal to $2 * (1 + \log N)$ in the remaining 13.5% of the successful cases. This previous work demonstrates that discovering QEAPs in a distributed manner within a P2P community is feasible. SPAD capitalizes on this earlier scheme, using it as a base for its QEAP discovery function.

The Internet Engineering Task Force (IETF) has proposed two different architecture models to provide and control the QoS of flows or classes of flows between two given hosts A and B on the Internet, namely the Integrated Service (IntServ) model [20] and the Differentiated Service (DiffServ) model [21]. The proposed SPAD system has a similar goal, as it aims to improve the end-to-end Qos between two peers within a community. IntServ proposes an approach whereby any necessary QoS resources are reserved on each network element (i.e. routers) on the Internet path between A and B. This resource reservation is done prior to the transmission of any data on the path, using the Resource Reservation Protocol (RSVP). However, the number of states to manage per data flow, and the complexity of RSVP both limit the scalability of the IntServ model. DiffServ addresses this scalability issue by confining the complexity (i.e. packet classification functions, and traffic conditioning functions) to the nodes at the network boundaries, and leaving only a simple task (i.e. differentiated forwarding policies of traffic aggregate) to the nodes inside the network core. However, the mapping of the user end-to-end QoS requirements into the differentiated forwarding policies is a difficult task. In addition to these individual limitations, IntServ and DiffServ also share the following drawbacks: (i) they both require some modifications in the network routers; and (ii) they both rely on the *default* end-to-end path given by the network routing mechanisms to implement their QoS-oriented schemes. As described in appendix A, in a significant number of cases this *default* path is not the optimal available path in terms of QoS (e.g. delay or reliability). In contrast, SPAD proposes a system that does not require any modifications in the core network elements, and that circumvents *default* Internet paths with poor QoS characteristics.

3. SPAD: a distributed architecture to discover QoS enhanced alternate paths

3.1. SPAD architecture overview

The SPAD architecture is based on a community of cooperating end-points that is organized as an

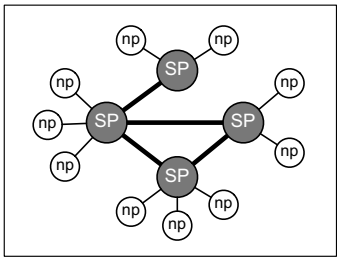

Fig. 3. A super-peer network (SP: super-peers; NP: normal-peers).

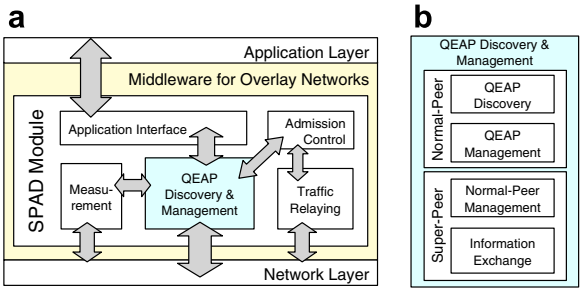

Fig. 4. (a) Structure of a SPAD module within a middleware framework on each SPAD peer and (b) Details of the "Discovery & Management module".

unstructured Super-Peer network [8]. When joining this network, an end-point would by default act as a normal-peer. Any peer may evolve to become a super-peer. Each super-peer is associated with a maximum number of normal-peers, as illustrated in Fig. 3 where this maximum number is equal to 3. In the SPAD community, super-peers are altruistic peers that in general have more resources at their disposal and therefore carry out more functions than normal-peers. Thus, in addition to performing the same functions as a normal-peers, a super-peer performs some extra functions for the community. Many research works investigate the design of distributed super-peer selection/election mechanisms [22], which remains a current research issue. The schemes from these contributions can be readily used in the SPAD architecture to incrementally construct scalable balanced super-peer topologies. For example in a basic selection scheme, each joining peer computes its *utility* based on its resources (e.g. computing, networking, storage), and communicates this value to its super-peer S. When S reaches its maximum number of manageable associated normal-peers, it selects its normal-peer N_X with the highest *utility*, and asks it to be a new super-peer. If N_X denies the request, S tries its next normal-peer with the highest *utility*. Otherwise, N_X becomes a new super-peer neighbor of S. It is then ready to accept association requests from new joining normal-peers.

The SPAD QEAP discovery method is based on the following basic scheme. Each peer collects a small amount of QoS information (e.g. delay) about its connectivity to other peers, and passes this information to its associated super-peer. Thus, the information about QoS parameters between the nodes on the overlay is distributed among the super-peers. When receiving a QEAP request from one of its associated normal-peers, a given super-peer will access this distributed knowledge and will return any information relevant to that request to the normal-peer. With this information, a normal-peer can discover potential QEAPs, and send queries to the corresponding candidate relay peers, requesting their participation in a QEAP. SPAD super-peers use two complementary information exchange schemes (a reactive one and a proactive one) to access the distributed QoS information. These schemes will be described in detail in Section 4.

Fig. 4 presents the structure of the SPAD module located within the middleware framework for overlay network running on each SPAD peer, as described in Section 1. Within this framework, the SPAD module could be associated with other modules providing services to the nodes of the overlay network. For example, the SPAD module could be associated with a transport module providing services similar or compatible with TCP. Such associations are not the focus of this paper, and will not be discussed further.

The *Measurement* module is responsible for collecting QoS information such as delay towards other peers. Measuring or estimating QoS parameters in an accurate, unobtrusive, and efficient manner is an open research issue. However recent advances in the measurement research area indicate that measurement techniques/applications with these properties are becoming available. In the case of end-to-end delay, an example of such an application is "King" [23]. It uses small numbers of cautiously crafted recursive DNS queries in a novel way to accurately estimate end-to-end delays. In the case of packet-loss rate, Allman et al. [24] propose a measurement technique, which does not require any traffic overhead. This technique is based on the count and analysis of retransmissions on a set of currently used TCP flows between two peers. It is assumed that the *Measurement* module within SPAD will be based on such techniques to efficiently

measure QoS information (such as delay and packet-loss rate) between any pair of peers in the community.

The *QEAP Discovery & Management* module is the primary focus of this paper. This module includes two sub-modules. All peers execute the *Normal-Peer* sub-module, in addition super-peers also execute the *Super-Peer* sub-module. The first sub-module handles the functions related to QEAP discovery, such as initiating QEAP requests, selecting the best QEAP among discovered ones, or processing queries to be a relay in other QEAPs. It also handles management functions for QEAPs currently being used by this peer, such as monitoring QoS on current QEAPs, recovering from QEAP failures. The second sub-module handles the super-peer related functions. It is responsible for the management of the associations between this super-peer and its set of normal-peers, namely setting or terminating associations, storing received QoS information, and processing QEAP requests. It is also the module that executes the two complementary schemes used to access the QoS information distributed among the SPAD community, in order to reply to QEAP requests.

The *Application Interface* module is responsible for making SPAD's operations transparent to the above application element. It provides programming primitives similar to the ones given by the current operating system network stacks (such as the Socket API in Unix based systems), allowing any applications to transparently use QEAPs. It also includes QoS mapping functions that allow applications that are aware of SPAD to request QEAPs with some specific QoS requirements.

The *Traffic Relaying* module handles the relaying of data traffic from a peer towards another peer. This relaying function is activated when the peer running this SPAD module accepts to be part of a QEAP between two other peers (similar to host *R* in Fig. 2). This module also monitors the available network resources for this peer, and provides this information to the *Admission Control* module.

The *Admission Control* module collects information from the other modules, and builds a view of the available computing and networking resources of this peer. Based on this view, it decides if this peer is able to participate as a relay in a given QEAP, and then passes this decision on to the QEAP Discovery module that will forward it to the requesting peer. The admission control policies involved in this decision process will be the subject of future studies.

3.2. Design requirement and assumptions

The performance of the proposed SPAD architecture depends on its ability to discover existing QEAPs, which in turn essentially relies on the performance of the reactive and proactive information dissemination schemes described in Section 4. These schemes should allow the retrieval of all existing information relevant to a given QEAP request, while minimizing the associated bandwidth usage and the latency. In that regard, the design of these schemes makes two assumptions, namely the quasi-symmetry and constancy of QoS parameters, and the similarity of external connectivity from within an Internet Autonomous System (AS). As described in Section 1, this paper presents a SPAD architecture that focuses on the discovery of delay QEAPs, i.e. alternate paths with better delay than default Internet paths. Without loss of generality, the remainder of this subsection shows the validity of the above assumptions in the context of delay QEAPs.

3.2.1. Quasi-symmetry and constancy of internet end-to-end delays

As explained in Section 3.1, each SPAD peer measures the delay from itself to a number of other peers and passes this information to its super-peer. The super-peers exchange this information, thus building an overall view of the community's delay characteristic. The accuracy of this overall view and the frequency of the measurement/information updates depend on the constancy of Internet delays. Based on the studies from [25,26,4], the SPAD architecture assumes that Internet delays are steady on time scales of 10–30 min.

Some measurement softwares, such as King [23], can be asymmetric: when running on a node A, they measure *delayAtoB*, but they do not provide *delayBtoA*. Furthermore, there is no guarantee that node B will measure *delayBtoA*. Thus this information might not exist within the community at a given time. However, it is possible to overcome this limitation and significantly minimize measurement overheads by assuming that *delayAtoB* is similar to *delayBtoA*. This approximation supposes that Internet end-to-end delays are quasi-symmetric despite IP routing mechanisms not guaranteeing symmetry of routing paths and related delays. Figs. 5 and 6 present the comparisons of *delayAtoB* versus *delayBtoA* for node pairs from the measurement datasets provided by [19,27]. Appendix A provides

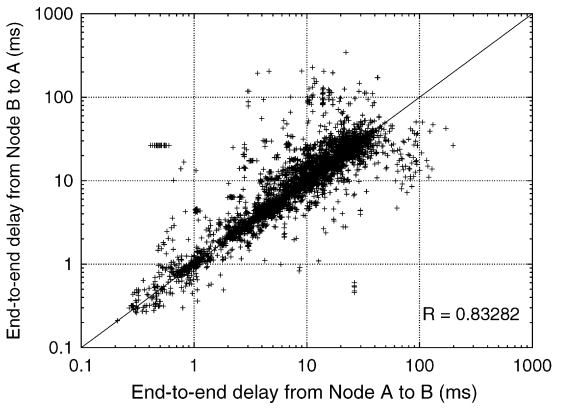

Fig. 5. Comparison between *delayAtoB* and *delayBtoA*. RIPE-TTM dataset: 25/05/04.

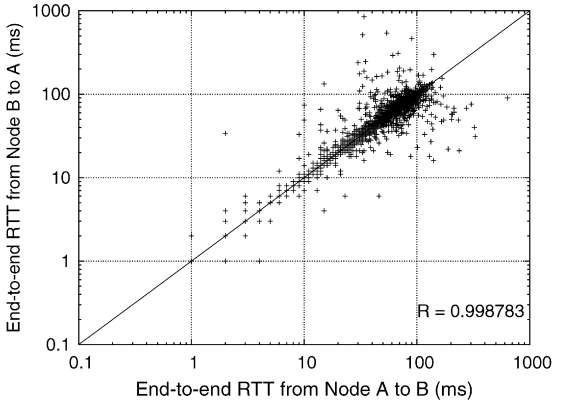

Fig. 6. Comparison between *delayAtoB* and *delayBtoA*. NLANR-AMP dataset: 30/01/03.

more details on these datasets. These figures show that the average ratios R between both delays are approximately 0.8328 and 0.9988, thus validating the above assumption.

3.2.2. Similar external connectivity of end-points within a same AS

The exchange of information messages within a large community of distributed peers consumes bandwidth. Minimizing this bandwidth consumption is a primary issue when designing an information exchange scheme. SPAD uses the following aggregation method to minimize the number and the size of these information messages. A simple *description* of delay between a host X_1 and another host Y_1 could be of the form: $[X_1; Y_1; delayX_1toY_1]$, similarly $[X_2; Y_2; delayX_2toY_2]$ for X_2 and Y_2. Assuming that two hosts within a given Autonomous System (e.g. hosts X_1 and X_2 within the same AS_U) would experience the same connectivity characteristics towards other hosts within a different AS (e.g. host Y_1 and Y_2 within AS_V), the above simple delay *description* can be aggregated into a complex entry of the form: $[AS_U; AS_V; delayUtoV; hostInU; hostInV]$, where $delayUtoV$ is the mean of $delayX_1toY_1$ and $delayX_2toY_2$; and where $hostInU$ would contain either X_1, or X_2. To allow some redundancy against host failure, $hostInU$ could contain a small list of hosts within AS_U. This complex entry provides aggregated delay information between AS_U and AS_V. Each SPAD super-peer performs this aggregation method on the delay information provided by their associated normal-peers. These aggregated entries constitute the information messages that are exchanged between super-peers, thus reducing bandwidth and storage utilization. Super-peers continuously update these entries based on the information from other peers. Each entry is marked with a timestamp to reflect its accuracy: the older an entry is, the less accurate it might be.

The proposed aggregation scheme would provide accurate results only under the assumption that hosts within a given AS experience the same connectivity characteristics towards hosts in other ASes. To support this assumption, we performed the following experiment on the measurement datasets from [28]. Again, appendix A provides more details on this dataset. The IP-to-AS mappings of the 177 hosts from [28] provide an average of 62($\pm$4.0) different ASes. Thus, on average 2.85($\pm$0.1) hosts are within the same AS. For each pair of AS (U; V), we computed the average delay from U to V, and the corresponding standard deviation. Fig. 7 shows that most AS pairs have their standard deviation of the delay between 0.01 and 10 ms. Fig. 8 indicates that for about 80% of the AS pairs, this standard deviation is equal or less than 6 ms. Although the studied datasets have a small number of ASes, these results still support the above assumption, and confirm the use of the described aggregation method.

Finally, the implementation of this aggregation scheme requires an accurate IP-to-AS mapping mechanism. Recent research studies have proposed various mechanisms to perform this mapping [29,30]. The RIPE RIS project [30] retrieves and compiles BGP routing tables 3-times a day from over 300 worldwide routers. Specialized RIS whois servers provide a mapping service of IP prefixes to AS numbers. Any host can query these servers: sending its IP address, and receiving back its AS

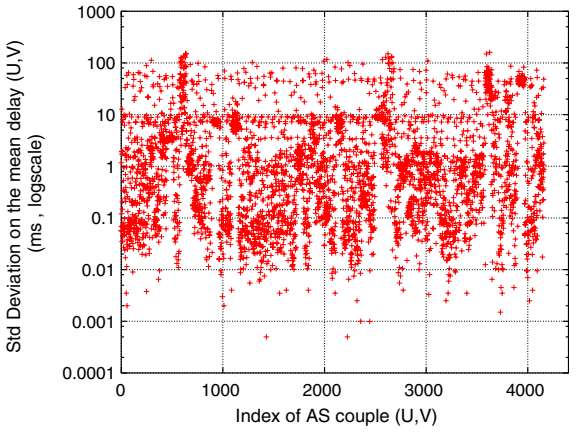

Fig. 7. Distribution of the standard deviation of the delay (U, V), for all AS pairs. PlanetLab dataset: 27/04/05.

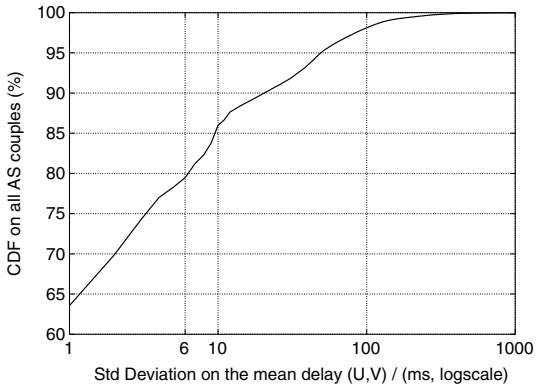

Fig. 8. CDF (on all AS pairs) of the standard deviation of the delay (U, V). PlanetLab dataset: 27/04/05.

number. In SPAD, peers joining the community will be able to use a similar mapping service to retrieve their AS number. They will add that information to their network address to compose their SPAD node ID. Since each peer makes a unique query, the additional load should not significantly impact the operations of servers designed to provide services to the Internet community.

4. Description of SPAD QEAP discovery schemes

4.1. SPAD peer initialization

When a new host N_A joins the SPAD community as a normal-peer, it performs the following initialization procedures. First, N_A establishes its partial knowledge of the community, by building a list L_A of other peers. This list is named a relay list (Rlist), and has a fixed-size. It contains a subset of the peers known to N_A that have the best QoS values, for example the lowest delay on the default Internet path from N_A to themselves. To build L_A, N_A contacts a small random set of super-peers $\{SP_i\}$, and requests their list of associated normal-peers. It measures/estimates the QoS parameter (e.g. delay) on the default Internet paths towards nodes within these lists, and includes in L_A the nodes corresponding to the paths with the best values (e.g. the lowest delay). To discover the $SP_i s$, N_A can use a scalable distributed directory service such as Chord [31]. For example, peers in the community could form a Chord-ring, N_A would join that ring, ask a small fixed number of its Chord successors for the identity of their super-peers, contact these super-peers and retrieve their list of associated normal-peers. Then it would evaluate the delay towards the nodes in these lists, and retain in L_A the nodes having the lowest delay. Thus L_A contains entries of the form $[N_A; N_X; delay N_A to N_X]$. If N_A and N_X are on the same AS (e.g. *myAS*), they would probably share the same gateway towards another node N_B not on *myAS*. The paths N_A to N_B and N_X to N_B would then be similar, and the chances that a QEAP exists from N_A to N_B via N_X would be low. Therefore to ensure node diversity in L_A, potential $N_X s$ send their AS number to N_A during its initial delay evaluations. A node N_X gets its AS number via the IP-to-AS mapping mechanism described in Section 3.2.2. N_A uses that information to filter out $N_X s$ on the same AS as itself, and $N_X s$ on redundant ASes. For example, if N_X, N_Y, N_Z have the same AS number, they are redundant, and N_A will consider only one of them for inclusion in L_A.

N_A periodically re-evaluates and updates the delays in L_A. Existing results on Internet delay constancy [26,25] can be used to select appropriate re-evaluation intervals. Furthermore, as N_A discovers other peers via user connection requests, it updates its list accordingly. With this updating strategy, the more a host participates in *overlay applications* by hosting a software element (Section 1), the more it can potentially be used as relay in QEAPs for other *overlay applications*. The Rlist size and the number of bootstrap super-peers are important scalability parameters. In SPAD, the list size is fixed to $\log \mathcal{N}$ ($\mathcal{N}$ being the number of peers in the community) and the number of bootstrap super-peers varies between 2 and 4. $\mathcal{N}$ is not known a priori and increases with new participating nodes.

However during initial deployment, the first participating peers could agree on an upper bound value of $\mathcal{N}$.

After building L_A, N_A randomly selects from the initial SP_is one super-peer S that is located in a different AS than itself. It associates with S, and uploads L_A to it. Using the aggregation procedure based on AS number described in Section 3.2.2, S processes the host-related delay information it receives from its set of associated normal-peers. As a result, S obtains aggregated AS-related delay information. It maintains this AS-related delay information in a Local Entry Table (LET). To keep a manageable load, super-peers have a maximum number of associated normal-peers.

4.2. SPAD reactive information exchange scheme

4.2.1. Scheme description

At the end of the above initialization phase, the SPAD module on N_A is ready to receive a QEAP discovery request from the application layer towards another peer N_B. When N_A's SPAD module receives such a request, it triggers the following Reactive Information Exchange Scheme. First, N_A evaluates the delay *delayDef* on the default Internet path to N_B, and constructs a QEAP request of the form [*requestID*; *src*; *dst*; *delayDef*; TTL_Q], with $src = AS_{NA}$, $dst = AS_{NB}$, and TTL_Q the requests time-to-live. N_A then passes that request to its super-peer S, which executes the query-processing algorithm described in Fig. 9. Using this algorithm,

```
NOTE:
a_request: <ID, from, to , delay, TTL>
an_entry: <from, to, delay, hostFrom, hostTo>
a_responseList: {entry1, …, entryN}

1 sp_search(a_request req) returns a_responseList
2 {
3  if (req.ID is in {alreadyProcessed_RequestID})
4     return null
5  add req.ID to {alreadyProcessed_RequestID}
6  for all entry E in this_superpeer.LET
8    if (E.from == req.from) or (E.from == req.to)
9        or (E.to == req.from) or (E.to == req.to)
10          add E to responseList
11 end // for all entry E ...
12 if ( req.TTL > 0)
13    req.TTL = req.TTL - 1
14    for all superpeer S in {connected_superpeer}
15      responseList=responseList+S.sp_search(req)
16    end // for all super-peer
17 end // if request.TTL
18 filteredResponse = filter(responseList, req)
19 return filteredResponse    }
```

Fig. 9. Pseudocode for the query-processing algorithm on SPAD super-peers.

```
1  filter(a_responseList list, a_request req)
                              returns a_responseList  {
2  for all entry E in list
3    if (E has a duplicate E' in list) remove E' from list
4    for all remaining entry G in list
5      if (E is of the form SRC/DST_to_X or X_to_SRC/DST)
6      and (G is of the form SRC/DST_to_Y or Y_to_SRC/DST)
7      and (X and Y are same AS number)
8          remove E from list
9    end // for all remaining entry G ...
10   if (E.delay > req.delay) remove E from list
11   for all entry F in this_superpeer.LET
12     if ( E is of the form SRC_to_X or X_to_SRC )
13         if (F is of the form X_to_DST or DST_to_X)
14             if ( E.delay + F.delay > req.delay )
15                 remove E from list
16     if ( E is of the form DST_to_X or X_to_DST )
17         if (F is of the form X_to_SRC or SRC_to_X)
18             if ( E.delay + F.delay > req.delay )
19                 remove E from list
20   end // for all entry F ...
21 end // for all entry E ...
22 return list    }
```

Fig. 10. Pseudocode for the filter algorithm on SPAD super-peers.

a super-peer selects all entries within its LET that match the request, in order to generate its part of the response (line 6–11). Then if the request's $\mathrm{TTL}_Q > 0$, it recursively forwards the request to all of its connected super-peers, and wait for their responses (line 12–17). Upon receiving the responses, it aggregates them with its own responses (line 15). It then executes a filtering process (line 18), before sending the result back to the requesting peer (line 19). As described on Fig. 10, the filtering process removes redundant (line 3, 5–8) or irrelevant entries from the aggregate response (line 10, 12–15, 16–19). Thus, responses to a QEAP request travel back along the forwarding path towards N_A. Super-peers on this path successively filter these responses, decreasing their size, and returning only relevant entries.

When the search process is complete, N_A receives a list of entries of the form [AS_X; AS_Y; *delay*-$AS_X toAS_Y$; *hostInX*; *hostInY*], with {AS_X or AS_Y} = {AS_{NA} or AS_{NB}}. N_A first analyzes this list to search for trivial QEAPs, namely cases where the list contains information such as $delayAS_{NA}toAS_Z$ and $delayAS_ZtoAS_{NB}$, with their sum smaller than *delayDef*. If such cases exist, N_A sends query messages to these potential relay nodes first (the host in the *hostInZ* field in the above example). Otherwise, it sends query messages in parallel to all the nodes {N_i} in the *hostInX* or *hostInY* fields. These queries ask a given node N_i to evaluate/measure the delay on the remaining hop $N_A toN_i$ or $N_i toN_B$. Each N_i checks its resource availability to participate in a QEAP from N_A. If sufficient resources

are available,[1] N_i responds to N_A with the requested delay evaluation/measurement, and keeps a temporary resource reservation, waiting for N_A's QEAP selection. Non-willing N_is return an infinite delay value to N_A.

When N_A receives back these delay values, it computes the overall delay on each alternate path. The paths with delays smaller than *delayDef*, are tagged as QEAPs. If such QEAPs exist, N_A selects the one with the lowest delay, and notifies the corresponding relay node N_{selected}, and N_B. N_{selected} commits the required resources and creates the necessary states to relay the traffic from N_A to N_B. N_A uses the QEAP and monitors the received QoS. Upon eventual QoS degradation, N_A can discover and use another QEAP. It is possible to optimize this operation by storing details of all the previously discovered QEAPs, using them as backup paths. At the end of the session, N_A notifies N_{selected}, and all the states and resources associated to the QEAP are released. The admission control function on each node N_i together with N_A's minimum delay path selection ensure simple load balancing among the candidate relay nodes.

4.2.2. Discussion on the reactive information exchange scheme

The above scheme is based on query flooding over an unstructured P2P system. Theoretical lack of scalability is the major limit of such technique. However, as discussed in [13], the use of a super-peer scheme significantly extends this limit. Furthermore, several studies [22] present design methods, query-processing algorithms, and topology construction protocols aiming at improving the scalability of super-peer systems. These techniques could be readily included in SPAD. Finally, other measurement studies suggest that unstructured P2P systems tend to self organize into power-law topologies [12]. In such topologies, node degrees follow a power-law distribution $P(k) \sim Ck^{-\alpha}$; with $P(k)$ the probability of a node to have k edges, α the power-law exponent, and C a normalizing constant. In [12], the authors show that networks with this topology characteristic scale more optimistically in regards to query flooding. Based on these studies, the experimentations on SPAD performances in Section 5 will only use power-law topologies to interconnect super-peers. Indeed, simulations based on other topologies such as pure-random or regular graphs, might not provide realistic results.

[1] This task involves an admission control function on the potential relay nodes, as illustrated in Fig. 4, but not detailed further in this paper.

4.3. SPAD proactive information exchange scheme

4.3.1. Scheme overview

The reactive scheme described in Section 4.4 uses a flooding-based technique, and is triggered by the reception of a QEAP request at a super-peer. Its performance, in terms of number of discovered QEAPs, is a function of the reach of the QEAP request, and will be described in Section 5.1. Assuming a fixed number of super-peers with a fixed minimum node degree, the reach of a request is a function of its TTL parameter (TTL_Q) [8]. Therefore, to increase the chances to discover QEAPs, a source node should set the TTL_Q of its QEAP requests to a high value. This would imply more message forwarding between super-peers, thus increasing the message cost and the overall search latency for a given QEAP request. To address this limitation, SPAD features a proactive scheme, which is complementary to the previously described reactive scheme, i.e. SPAD super-peers execute both schemes within their *QEAP Discovery* module. The proposed reactive scheme allows super-peers to perform an information dissemination algorithm based on probabilistic flooding during their idle time. This algorithm allows the exchange of connectivity information, namely LET entries, between super-peers. As a result, a given super-peer would potentially store local copies of all the existing connectivity information relevant to its associated normal-peers. A QEAP request from a normal-peer would then need to be forwarded to only a few hops among super-peers (ideally one or two) in order to generate the same amount of relevant responses as in a SPAD system featuring only the previous reactive scheme. This allows the use of lower TTL_Q values in QEAP requests, thus resulting in a lower per-request message cost and search latency. The exact number of forwarding hops depends on the achieved dissemination state of the proposed algorithm, and is discussed further in the next subsections.

4.3.2. Probabilistic flooding and power-law network

The simplest information dissemination strategy is the flooding technique. Unfortunately, this strategy requires significant bandwidth for large dynamic networks [32], which limits the scalability

of the system. As introduced in Section 2, there exist other dissemination strategies that achieve full propagation at a lower bandwidth cost than classic flooding. Due to the fact that the probabilistic flooding strategy does not require any extra group/membership management functionality, it is an adequate candidate for SPADs information dissemination scheme.

Probabilistic flooding derives from the bond-percolation theory [16]. In this scheme, a node forwards newly generated or received information to its direct neighbors with a probability p, and drops that information with the complementary probability $(1-p)$. It also drops already processed or forwarded information (i.e. *infect-and-die* policy per new information). Results from the bond-percolation theory demonstrate that for a given connected network graph, there exists a threshold dissemination probability P_C at which the information reaches all nodes at a minimum message cost. When this threshold is reached, the network is reduced to a minimum connectivity state that provides complete reachability without any redundant paths (i.e. the information does not reach the same node twice). In [18], the authors studied the application of that scheme to search (i.e. request forwarding) in power-law networks. They provided an analytic expression of P_C based on the power-law exponent α of the network being considered. Furthermore, depending on α, they showed that P_C could be smaller than 1. However their expression assumed infinite request/information TTL. Extending their study, we analyzed the dissemination probability with respect to the information propagation TTL and the amount of reached nodes. We used BRITE [33] to generate 10^3 power-law topologies based on the Barbasi-Albert model (with α between 2.8 and 3). Then for each topology, we randomly selected a node from which we started the propagation simulation.

Figs. 11 and 12 show the percentage of nodes reached by the information for different values of dissemination probability (P_D), dissemination TTL (TTL_D), number $\mathcal{N}$ of super-peers, and minimum node degree m. The comparison of these figures confirms the following two propositions: (i) for a given network, propagation reach depends on both P_D and dissemination TTL_D; (ii) higher minimum node degree allows faster information propagation. From Fig. 12, we observe that the network diameter is 4 for power-law topologies with 100 nodes and $m = \{3;4\}$. When $\mathrm{TTL}_D = 4$ in a classic flooding strategy (i.e. $P_D = 1$), the information reaches

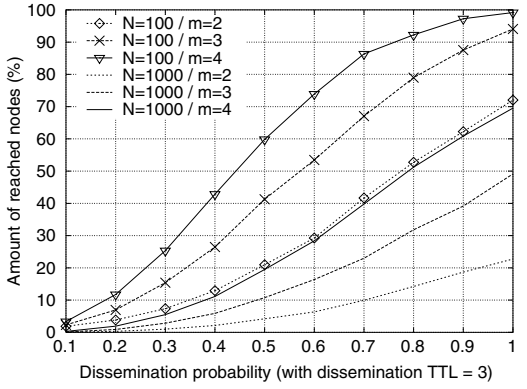

Fig. 11. Amount of reached nodes for different dissemination probabilities and a TTL = 3.

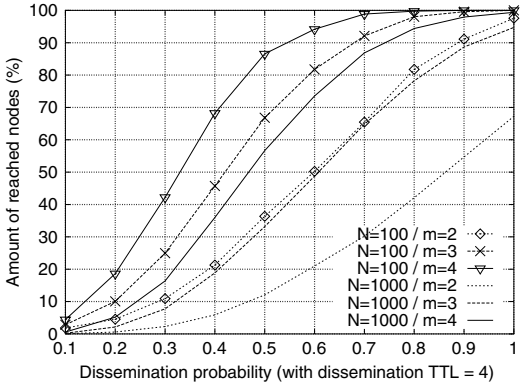

Fig. 12. Amount of reached nodes for different dissemination probabilities and a TTL = 4.

100% of the nodes. However, one can notice that the information has already reached almost all the nodes with a dissemination probability of $P_D = 0.6$ for $m = 4$, and $P_D = 0.8$ for $m = 3$. The same behavior is also apparent for the other topologies (not presented in this paper). These results demonstrate that in power-law networks, an information dissemination strategy using probabilistic flooding can achieve near full propagation at a lower message cost (20–40% less in the previous example) and a similar dissemination TTL compared to a classic flooding strategy. These results do not allow any conclusions on the scalability of probabilistic flooding. This is the subject of ongoing research studies [18].

4.3.3. Selection of P_D and TTL_D

The choice of P_D and TTL_D determines the information reach, and is a function of the characteristics

of the super-peer network. This subsection presents an analytical study on the selection of these parameters. In [34], the authors provided a brief study of the average distance between any two nodes in random graphs. The following analysis is based on their problem formulation.

For simplicity, we consider a graph of $\mathcal{N}$ nodes, with a fixed node degree (that we denote by $M+1$) equal to 3. The following analysis remains valid for graphs with arbitrary node degree distribution, such as power-law distribution ($M+1$ would then represent the average node degree). Fig. 13 shows this graph, where each node represents a SPAD super-peer. We select node A as the dissemination source of a new piece of information. We can then represent the graph as a tree structure (Fig. 13); where A is the root (rank 0), and where B, D, E (A's direct neighbors) are one level below (rank 1). Following the graph's topology, we add each node's direct neighbors in the tree until all nodes are visited, with the only constraint that a tree node at rank k cannot have its parent (rank $k-1$) as one of its direct children (rank $k+1$). We note that a given node from the graph can appear several times within the tree. For example, node C appears for the first time at rank 2, then many times at rank 3. This is due to the fact that a node has several neighbors in the graph; hence it could be the child of several parent nodes in the tree. Furthermore, we also note that A has $M+1$ children nodes, and any node at rank k, with $k>0$, has M children nodes.

In the next step of our analysis, we number the nodes according to their sequence of inclusion in the tree. For example, node A has the number 1, node C has the numbers 8, 12, 21. We denote by t_k the last node number at rank k. In a classic flooding strategy, we have:

$$t_1 = 1 + (M+1), \quad \text{and}$$
$$t_k = 1 + \sum_{i=0}^{k-1} (M+1)M^i. \tag{1}$$

However, in a probabilistic flooding strategy with a probability p, we have:

$$t_1 = 1 + (M+1)p, \quad \text{and}$$
$$t_k = 1 + \sum_{i=0}^{k-1} (M+1)M^i p^{i+1} \tag{2}$$

using iterative simplification on (2) and fixing $Mp \neq 1$, we obtain:

$$t_k = \frac{(M+1)M^k p^{k+1} - (p+1)}{(Mp-1)}. \tag{3}$$

We denote by $f(t)$ the total number of unique nodes in the tree after the tth node (N_t) has been added. For example, in Fig. 13, $f(4)=4$, and $f(10)=7$. Then $f(t_k) - f(t_{k-1})$ represents the number of unique nodes in the tree at rank k. We define δ_t:

$$\delta_t = f(t) - f(t-1) \tag{4}$$

with $\delta_t = 1$ if N_t is a new node, and $\delta_t = 0$ if N_t has been previously added to the tree. We denote by $P(\delta_t = 1)$ the probability that N_t is a new node. Prior to the addition of N_t, there are by definition $f(t-1)$ unique nodes in the tree. Thus there are $(\mathcal{N} - f(t-1))$ unique nodes which are not yet in the tree. Assuming a uniform random selection of N_t, we have:

$$P(\delta_t = 1) = \frac{\mathcal{N} - f(t-1)}{\mathcal{N}}. \tag{5}$$

We denote by $E(t)$ the expected value of $f(t)$, and we take the expectation on both side of Eq. (4), using the result (5). We obtain:

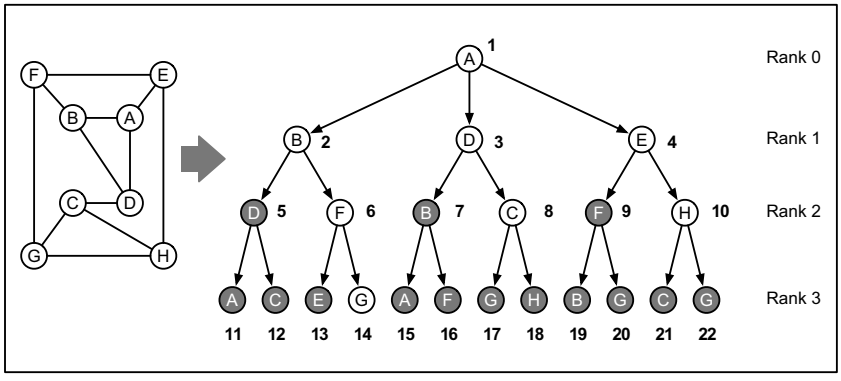

Fig. 13. Graph and tree examples for the theoretical analysis of SPADs information dissemination strategy.

$$E(t) - E(t-1) = \frac{\mathcal{N} - E(t-1)}{\mathcal{N}}$$
$$\Rightarrow E(t) = 1 + \frac{\mathcal{N}-1}{\mathcal{N}} E(t-1) \quad (6)$$

then we solve (6) iteratively (fixing $f(0) = 0$) to obtain:

$$E(t) = \mathcal{N}\left(1 - \left(\frac{\mathcal{N}-1}{\mathcal{N}}\right)^t\right), \quad (7)$$

which represents the expected value of the total number of unique nodes in the tree after the inclusion of the tth node. From this result, we can estimate the number of unique nodes reached by a piece of information that was issued by node A and was disseminated through the graph using a probabilistic flooding strategy. Indeed for a given pair of dissemination probability P_D and time-to-live TTL_D (corresponding to p and k, respectively), expression (3) gives a node number $t_k = \mathrm{TTL}_D$, that we can use in expression (7) to obtain an estimation of the mean number of unique reached nodes. Therefore, in the proposed SPAD information dissemination scheme, if we assume that a given super-peer S knows $\mathcal{N}$, the maximum number of super-peers,[2] and that it can estimate the average number of direct neighbors $M+1$; then using (3) and (7) it can compute estimates of P_D and TTL_D that would allow its information to reach, at a minimum message cost, a fraction x (as close to 1 as possible) of all the super-peers.

For instance, using (7) and resolving $E(t) = \mathcal{N}$, a super-peer S would obtain a node number t. It would then use this number t into Eq. (3) to obtain a couple (P_D; TTL_D). Eq. (3) can provide multiple pairs of (P_D; TTL_D) that achieve similar dissemination reach, but with different incurred message cost. For example in Figs. 11 and 12, the pairs ($P_D = 0.5$; $\mathrm{TTL}_D = 3$) and ($P_D = 0.36$; $\mathrm{TTL}_D = 4$) both provide a reach of 60% for a network with $N = 100$ and $m = 4$. In such cases, S uses Eq. (10) given in Section 4.3.5 to compute the message cost incurred by each pair, and selects the pair having the lowest message cost.

To verify the model provided by (3) and (7), we generated 10^3 BRITE power-law topologies of $\mathcal{N} = 10^3$ nodes with a minimum node degree $m = 3$, and performed probabilistic flooding simulations with different P_D and TTL_D. For these topologies, we measured an average network diameter of 5, and an average $M + 1 \approx 5.92(\pm 6.77)$. For $\mathrm{TTL}_D = 5$, Fig. 14 presents the comparison of the experimental propagation on the generated topologies and the theoretical propagation implied by (3) and (7), using the measured value of $M+1$. According to these results, the estimation given by (3) and (7) of the average numbers of nodes in the network at the kth rank (e.g. $k = 5 = \mathrm{TTL}_D$) is slightly smaller than the experimental values. Therefore, when using (3) and (7) to determine optimal values of P_D and TTL_D, a SPAD super-peer will obtain conservative values. As presented on Fig. 14, using $P_D = 0.6$ and $\mathrm{TTL}_D = 5$ will theoretically guarantee a reach of at least 60%; which is confirmed by the experiments showing an average reach of slightly less than 80%.

4.3.4. Integration of probabilistic flooding in SPAD

The previous subsection provides an analytical model (Eqs. (3) and (7)) to dynamically compute P_D and TTL_D. We use this model in the following super-peer bootstrap mechanism to include the probabilistic flooding scheme into SPAD. The accuracy of this model depends on the value of $M+1$. In a stationary topology, where the number of super-peers remains constant after a given time, $M+1$ can be analytically computed using the topology's power-law distribution. This computation is described in appendix B. However, in a dynamic P2P network, where super-peers join and leave randomly, each super-peer needs to dynamically estimate the value of $M+1$. The following bootstrap mechanism, based on the strong law of

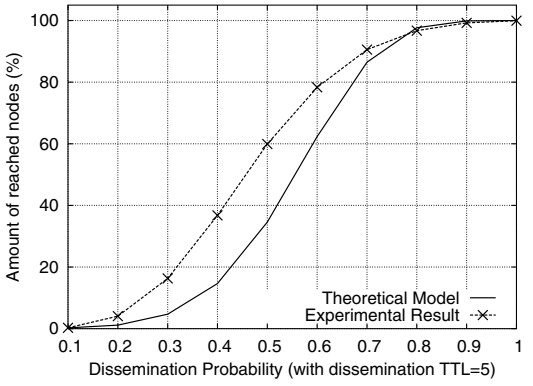

Fig. 14. For $\mathrm{TTL}_D = 5$, comparison of information dissemination on experimental topologies versus theoretical model implied by (3) and (7).

[2] $\mathcal{N}$ could be a system-wide parameter, set during initial SPAD deployment and passed-on to new super-peers during their bootstrapping phase.

large numbers, allows such an estimation. A new SPAD super-peer S_X retrieves the maximum number $\mathcal{N}_S$ of allowed super-peers in its SPAD community (e.g. from an already existing super-peer such as one of its neighbors). It also measures m_{SX}, the number of its direct neighbor super-peers. Then it communicates its measured value m_{SX} to all of its direct neighbors S_Is, and receives back their measured values m_{SI}s. Finally, S_X takes the average of its m_{SX} and all the received m_{SI}s as its estimation of $M+1$. S_X uses the values of $\mathcal{N}_S$ and $M+1$ as input parameters into the expressions (3) and (7). This bootstrap mechanism allows a new SPAD super-peer S_X to dynamically compute a couple (P_D; TTL_D) that theoretically ensures a complete dissemination of its information among the other super-peers. Furthermore upon receiving the value m_{SX} from S_X, each S_I updates its estimation of $M+1$, and recalculates its parameters P_D and TTL_D. This allows dynamic updates of these parameters as the topology of the super-peer network evolves.

After becoming a super-peer and completing the related initialization tasks (e.g. association with normal-peers and LET construction, as described in Section 4.1), a node S uses its idle time to perform the functions associated with the dissemination scheme. S is idle whenever it is not processing a QEAP request. As mentioned earlier, the entries in the LETs of super-peers are the information to disseminate. The first step for S is to communicate its LET to its directly connected neighbors. S initially sends its entire LET. When its LET subsequently changes due to updates from associated normal-peers, S will send only the modified entries to its neighbors. LET propagation is done using the probabilistic dissemination technique (Section 4.3.2). The second step is the processing of incoming LET entries, as described in Fig. 15. After receiving an incoming entry E, S verifies that it has not already processed it earlier (line 3–7), and adds it to the list of entries to forward via the dissemination scheme (line 8). Then S compares E with the entries from its own LET (line 9–13). If E matches one of these entries, then it contains information potentially relevant to future QEAP requests from S's associated normal-peers. Therefore, S retains E and stores it in a Remote Entry Table (RET). S also compares E to its existing RET entries, to determine if E could potentially provide more information to one of them (line 14–19). The whole matching procedure (line 9–19) aims at selecting only incoming

```
NOTE:
    an_entry: <ID, ASFrom, ASTo, delay, hostinFrom, hostinTo>

1   process_incoming_entry (an_entry E)
2   {
3    if (E.ID is in {already_processed_entry_ID})
4       discard E and exit
5    if ({already_processed_entry_ID}.size > threshold)
6       discard {already_processed_entry_ID}.older_element
7    addE .ID to {already_processed_entry_ID}
8    addE  to {entries_to_forward_via_dissemination_scheme}
9    for all entry F in this_superpeer.LET
10      if ((E.ASFrom == F.ASFrom) || (E.ASFrom == F.ASTo)
11          || (E.ASTo == F.ASFrom) || (E.ASTo == F.ASTo))
12          add E to this_superpeer.RET
13   end // for all entry F
14   for all entry G in this_superpeer.RET
15      if ((E.ASFrom == G.ASFrom) && (E.ASTo == G.ASTo))
16          add E.hostinFrom to the list G.hostinFrom
17          add E.hostinTo to the list G.hostinTo
18          update G.delay and exit
19   end // for all entry G
```

Fig. 15. Pseudocode of the Incoming Entry Processing algorithm executed by SPAD super-peers.

entries that contain information related to S's associated normal-peers. S does not propagate its RET entries to its neighbors. Indeed, via LET propagation, they may already have received the raw information on which they would probably have made a different selection than S. As dissemination progresses, S would store locally more and more entries (from further super-peers) relevant to its associated normal-peers. Each super-peer fixes the size of its own RET based on its available resources.

When S receives a QEAP request from N_A (one of its normal-peers) towards another node N_B, it processes the request as described in Section 4.2.1 with the following two modifications. First, S includes its RET entries in the selection process that builds its part of the response R_S. Therefore, its part of the response R_S contains a set of matching entries from both its LET and RET, these entries are of the form [AS_U; AS_V; $delayUtoV$; $hostInU$; $hostInV$] where {U or V} is the AS of either node N_A or N_B. The second modification concerns the request forwarding to other super-peers. Depending on the state of the dissemination process, S might already have in its RET all the existing information relevant to the request. In such case, S would not need to forward the request to its neighbor super-peers. However, due to the dynamic character of the peer-to-peer SPAD community, and the probabilistic nature of the dissemination scheme, this ideal case might not be reached. Therefore, S might still need to forward the request, but with a lower request TTL (TTL_Q) than the forwarding process of Section 4.2.1. We propose the following simple process to make such decision. S constructs $ListAS_S$, the list of distinct ASes in its LET and RET. Then $|ListAS_S|$ is a coarse lower bound of

the total number of ASes involved in the SPAD community. Based on $ListAS_S$, a coarse lower bound of the number of potentially relevant response entries in the community is given by:

$$NumResp_S = \begin{cases} 2*(|ListAS_S|-2) & \text{if (AS of } N_B) \in ListAS_S, \\ 2*(|ListAS_S|-1) & \text{otherwise.} \end{cases} \quad (8)$$

Then:

- if $|R_S| \geqslant \beta * NumResp_S \Rightarrow S$ decides that its local part of the response already contains enough entries, and does not forward the request
- else S forwards it with a

$$\mathrm{TTL}_Q = \begin{cases} \gamma * \lceil NumResp_S/|R_S| \rceil & \text{if } |R_S| \neq 0, \\ \gamma & \text{if } |R_S| = 0, \end{cases}$$

with $0 < \beta \leqslant 1$ and $\gamma \in \mathbb{N}^*$, a couple of system-wide parameters (e.g. $\beta = 0.7$ and $\gamma = 3$). In our future work, we plan to develop another technique based on Markov decision processes to evaluate with more precision the state reached by the information propagation. The remainder of the request processing, the response processing, and the QEAP set-up and utilization are identical to the descriptions in Section 4.2.1.

4.3.5. Discussion on the proactive information exchange scheme

Section 4.3.1 claimed that the proposed proactive scheme provides gains in search message cost and latency. One could argue that these gains are balanced by the dissemination message cost and latency, thus implying no real improvement to SPAD. This subsection proposes a simple analysis to address this concern. A normal-peer N_A issues a QEAP request towards its associated super-peer S_1. We assume that there exists a piece of information J that is relevant to this QEAP request. Then J is necessarily stored on S_X, one of the existing super-peers, and the probability of finding J is $P = 1/\mathcal{N}_S$, with $\mathcal{N}_S$ being the number of super-peers. Without the proposed information dissemination scheme in SPAD, N_A would successfully access J only if its QEAP request reaches S_X. Using the QEAP processing reactive scheme from Section 4.4, this would incur an average message cost among super-peers of:

$$C_1 = (M+1)\sum_{k=0}^{\mathscr{D}} M^k \quad (9)$$

with $M+1$ being the average super-peer degree, and $\mathscr{D}$ being the network diameter of the SPAD overlay network. In a SPAD community with the proposed information dissemination scheme, J would be disseminated from S_X to all the remaining super-peers. It would eventually reach S_1, with a message cost between super-peers of:

$$C_2 = p(M+1)\sum_{k=0}^{\mathrm{TTL}_D} p^k M^k. \quad (10)$$

According to the analysis and results of Section 4.3.3, $p < 1$ and $\mathrm{TTL}_D \leqslant \mathscr{D}$. Therefore, the proposed proactive scheme provides a gain in term of message cost between super-peers.

We define $C_3 = K\mathscr{L}$, the average latency cost to access J, with $\mathscr{L}$ being the average latency on a hop between two super-peers, and K being the number of hops required to reach S_X from S_1. Without the dissemination scheme, we have $K = \mathscr{D}$, whereas with the dissemination scheme, J would gradually reach super-peers closer to S_1 (and eventually S_1 itself), resulting in $K \leqslant \mathscr{D}$. Therefore to access J, the proposed proactive scheme may provide a gain in latency, depending on the advancement of the information dissemination process. Since super-peers perform the proactive information exchange scheme as a background task, we do not consider the dissemination latency in this analysis.

Furthermore, the information gathered by a super-peer via the proposed dissemination algorithm could potentially be used to reply to multiple QEAP requests from its associated normal-peers. In the above example, once J arrives at S_1, it could not only be part of the response to N_A's QEAP request, but it could also be part of the responses to subsequent QEAP requests issued by S_1's associated normal-peers, hence a further gain in message cost and latency. Finally, as stated earlier due to the dynamic behavior of SPAD's peer-to-peer network and the probabilistic nature of the dissemination algorithm, a state of complete information dissemination might not be achievable among SPAD super-peers. However even with partial dissemination, the evaluations in Section 5.2 show that the proposed scheme still provides significant gains.

4.4. Integration of loss-QEAP search in SPAD

The previous SPAD schemes were designed to support the integration of new QoS parameters in their QEAP discovery mechanisms with minimal

modifications to the overall system. To illustrate such integration, this subsection presents the set of required modifications, which allow the SPAD system to discover QEAPs with enhanced packet-loss rate.

First, this integration requires the addition of a packet-loss measurement mechanism to the SPAD component within each peer. As introduced in the *Measurement* module description in Section 3.1, Allman et al. [24] propose a technique based on TCP retransmission analysis to estimate end-to-end packet-loss rates in an accurate and unobtrusive way. It is assumed that the SPAD system with packet-loss rate capability will be able to use such technique within its *Measurement* module.

Second, Zhang et al. [25] showed that end-to-end packet-loss rates are constant on a scale of less than 10 min. This is lower than the 10–30 min constancy of end-to-end delay, as mentioned in Section 3.2.1. Therefore, the SPAD mechanism responsible for triggering measurement updates and their propagation to the super-peers needs to account for this difference in constancy durations.

Third, the computation of the value of the QoS parameter over an entire QEAP is different for the delay and the packet-loss rate parameters. In the case of the delay, the overall latency is the sum of the single delays on the component paths, i.e. delay is an additive metric. However for the packet-loss rate parameter, the computation of the overall value is performed using the complement of the single packet-loss rates, which is a multiplicative metric [7]. For example, if Q_{AXB} is a QEAP from hosts A to B via a relay host X, and P_{AX} and P_{XB} are the packet-loss rates on the respective paths $[A \rightarrow X]$, and $[X \rightarrow B]$, then P_{AXB} the packet-loss rate on Q_{AXB} is defined as follows:

$$P_{AXB} = 1 - (1 - P_{AX}) * (1 - P_{XB}). \quad (11)$$

When searching for a QEAP between two peers, the modified SPAD discovery schemes will use this expression (11) to compute the overall packet-loss rates on the potential alternate paths. It will then compare the resulting values with the packet-loss rate on the default path to determine if a given alternate path is a QEAP.

Finally, given the added capability of discovering QEAPs that provide enhanced delay and/or packet-loss rate, a SPAD peer N_A will have a larger number of discovered QEAPs that it can potentially use to forward the traffic of one of its application $\mathscr{A}$. The selection of the *best* QEAP depending on the provided delay gain, packet-loss gain, and the QoS requirements of the application $\mathscr{A}$ is important, as it determines the user's QoS perception. Such QEAP selection mechanisms are important components of a complete SPAD system. The design and evaluation of such schemes are currently under investigation and will be presented in future works.

5. SPAD performance evaluation

To evaluate the proposed SPAD QEAP discovery schemes, we used trace-based simulations on the delay measurement datasets (described in appendix A) from the NLANR-AMP, and the PlanetLab All Pair-Ping projects [27,28].

5.1. Evaluation of the reactive information exchange scheme

We performed the following experimentations to evaluate the performance of the SPAD reactive scheme. From the available 107 nodes in the NLANR-AMP dataset [27], we randomly selected 10 nodes to perform as super-peers, and evenly associated to them the remaining 97 nodes as normal-peers. The size of the Rlist was set to 6 on each normal-peer. We used topologies from BRITE [33] to connect the super-peers in a power-law topology based on the Barbasi-Albert model. The following results are averaged over 10^4 trials.

Fig. 16 presents a performance comparison between SPAD, the former 2-level QEAP discovery scheme (P2LS) [4], introduced in Section 2, and a Random search scheme (RAND) where a node looking for a QEAP queries a fixed-size list of random potential relay nodes. The Performance axis

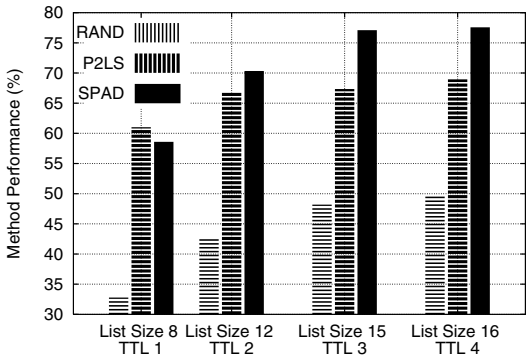

Fig. 16. Performance comparison between SPAD reactive scheme and other QEAP discovery schemes. NLANR-AMP dataset: 30/01/03.

corresponds to the percentage of existing QEAP that are discovered by each method. Starting with SPAD, we varied TTL_Q from 1 to 4 (4 being the diameter of our experimental super-peer network). We then noted the average size of the response list for each TTL_Q category, and used that value to set the size of the candidate relay list in the P2LS and the RAND schemes. For a $TTL_Q > 1$, the SPAD reactive scheme significantly outperforms the 2 other methods, discovering 77.58% (±0.5726) of existing QEAP for $TTL_Q = 4$. The performance of SPAD is a function of the reach of a query, i.e. the more super-peers a query reaches, the more relevant information will be received by the source node, and the more potential relay nodes will be known to the source. Yang and Garcia-Molina [8] established that the reach in a power-law super-peer network is a function of the existing number of super-peers, their average node degree, and the requests TTL_Q. They provided an extensive study of the influence of these parameters on the scalability of such networks. Table 1 presents the search cost in terms of messages being exchanged between super-peers, and latency related to these exchanges. The latency cost does not include processing delay within super-peers. As expected, both costs increase with TTL_Q. The latency associated with the chance of discovering 77.10% of existing QEAPS is still close to 200 ms. For several applications, such as VoIP, we believe that it is a reasonable connection set-up time, since the resulting QEAP would probably be used for several minutes, and would provide better delay than the default path for that duration (more details on QEAP delay gains in appendix A). Table 1 also shows the average size of responses generated by a request for a version of SPAD with response filtering on super-peers (as described in Fig. 16) and a version of SPAD without filtering. Using response filtering on super-peers significantly decreases the size of responses sent back to the originator of a request, thus decreasing the associated bandwidth cost.

Table 1
SPAD performance characteristics for different TTL_Q

TTL	SPAD performance (%)	Search cost (message number)	Search cost latency (ms)	Response list size (number of entries)	
				With filter	Without filter
1	58.59	2.85	97.6	8.26	11.18
2	70.33	6.12	160.8	11.98	17.22
3	77.10	9.28	208.4	15.22	22.75
4	77.58	10	222.1	15.87	23.86

5.2. Evaluation of the proactive information exchange scheme

We performed the following experiments to evaluate the performance of the SPAD proactive scheme. We built the simulation networks by randomly selecting 15 and 30 super-peers (i.e. $\mathcal{N}_S = \{15 \text{ or } 30\}$) among the available 177 nodes in the PlanetLab dataset [28], and by randomly assigning the remaining nodes as associated normal-peers. This resulted in about 11 and 5 normal-peers per super-peer. Using BRITE [33], we organized the super-peers in a power-law topology. We built all the Rlists (with a fixed-size of 10) and the LETs, and performed information dissemination with different target reaches. In an ideal case, where the set of super-peers remains constant and where the proactive scheme has been operating for a sufficient duration, a given information (e.g. a LET entry from a super-peer) would reach all the super-peers. This case corresponds to an achieved 100%-reach. However, due to the dynamic behavior of peer-to-peer networks, such a 100%-reach state might not be achieved at a given time. Therefore, to evaluate the performance of the proposed proactive scheme in these non-ideal cases, we specifically set the reach of the information to some given target values in the following experiments. We used the model from Section 4.3.3 to determine the P_D and TTL_D corresponding to each target reach. Finally, at a random timestamp, we selected random nodes as the source and the destination of a QEAP request, and started the search process for different query TTL (TTL_Q). We repeated this experiment 10^4 times. Fig. 17a and b present the result for $\mathcal{N}_S = 15$($\mathcal{N}_S = 30$ provides similar results). For a $TTL_Q = \{1, 2, \text{ or } 3\}$, a SPAD architecture with the information dissemination scheme allows the discovery of more QEAPs than a SPAD architecture without this scheme. Moreover, for a given TTL_Q, a higher dissemination reach provides a higher number of discovered QEAPs. Assuming that super-peers initially set P_D and TTL_D to aim at 100%-reach, starting from a state of 0%-reach, dissemination reach will increase as time progresses and super-peer network remains quasi-stable. For example with $TTL_Q = 1$ and a minimum node degree $m = 3$, a dissemination reach of 50% (i.e. the LET information of each super-peer has reach

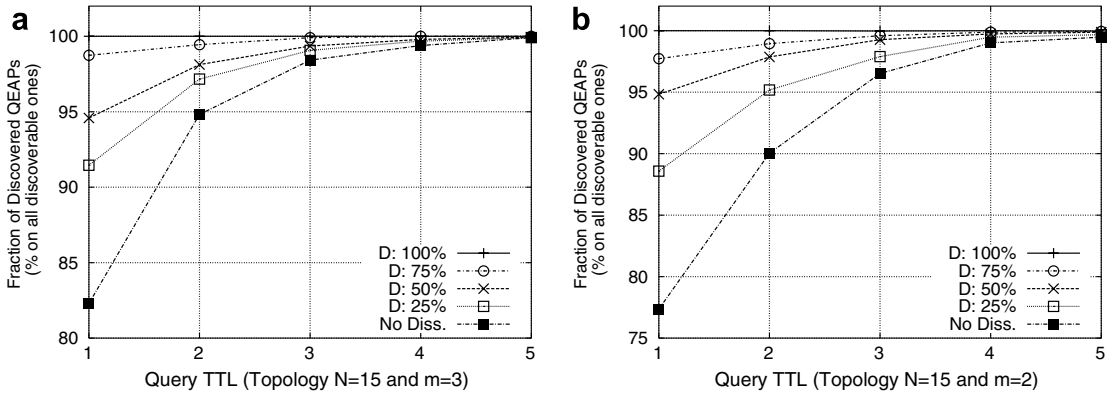

Fig. 17. Fraction of discovered QEAPs among all the discoverable ones, for different TTL_Q and information dissemination reach D. Topology with $\mathcal{N}_S$ =15 super-peers and minimum node degree $m = 3$ (a) and $m = 2$ (b).

50% of all the super-peers) allows the discovery of about 95% of the discoverable QEAPs,[3] while only about 82.5% are discovered without the dissemination scheme (labeled No Diss. on the figures). This difference increases as the minimum node degree of the network decreases. On Fig. 17, for $TTL_Q = 1$ and $m = 2$, the difference between the two scenarios (i.e. 50%-reach and No-Dissemination) in terms of discovered QEAPs is about 17.5%, compared to the 12.5% for $m = 3$. Indeed, the more connections a super-peer has, the more other super-peers a request will reach at each TTL_Q steps (i.e. the more information it will access), hence the decrease in the performance difference between 50%-reach and No-Diss. scenarios.

For one QEAP request processing, Table 2 presents the average number of messages generated by super-peers for different values of TTL_Q. Table 3 shows the average QEAP search latency depending on the request TTL_Q. This latency takes into account the request forwarding delay, and the response delay on the reverse path. It does not account for processing time on super-peers. It is only a function of the number of hops on the corresponding request path, and does not depend on network topology. As expected, Table 2 shows that when $TTL_Q = 5$ (the network diameter), the message costs stabilize just under the numbers of super-peers ($\mathcal{N}_S = 15$ and $\mathcal{N}_S = 30$). Indeed, once a query has reached all existing super-peers, it cannot be forwarded anymore. The steep increase in message cost for initial TTL_Q values is due to the power-law nature of the topologies, i.e. requests reach a highly connected super-peer after one hop, which results in higher message generation at the next TTL step.

These tables and figures provide sufficient information to quantify the message cost and search latency gains of the proposed information dissemination scheme. For example, if we assume a SPAD community with $\mathcal{N}_S = 15$ and $m = 3$ and sufficient initial time to have a 50%-reach dissemination, the information dissemination scheme allows the discovery of around 95% of the discoverable QEAPs at an average cost of 6.2 messages and a latency of 190.9 ms. Achieving a similar performance without the proposed scheme requires 12.1 messages, and a latency of 380.5 ms.

5.3. Discussion on SPAD experimental evaluation

The experiments presented in this paper were all based on simulations using delay measurements between real Internet hosts. These measurements provide a realistic snapshot of Internet delay characteristics, thus ensuring that our simulation environments accurately mimic the characteristics of a real-world environment. However, these simulations do not account for dynamic changes in network connectivity. Indeed for each trial, the simulated environment is not modified to reflect the changes due to the resources being used by any previously discovered QEAPs. To address this issue, and perform SPAD evaluations in a dynamic environment, we are currently developing a SPAD

[3] A QEAP is discoverable (i.e. can be discovered) if there exists a SPAD super-peer that holds some information about it. Within a given SPAD community at a given time, it is possible that some existing QEAPs could not be discovered because there are no super-peers that store information about them.

Table 2
Search cost in messages for different TTL_Q (i.e. TTL of the QEAP request)

	Total number of messages generated by super-peers at each TTL_Q step				
	Values of TTL_Q				
Network topology	1	2	3	4	5
$\mathcal{N}_S = 15$; $m = 2$	4.6	10.3	13.1	14.3	14.8
$\mathcal{N}_S = 15$; $m = 3$	6.1	12.1	14.0	14.7	14.9
$\mathcal{N}_S = 30$; $m = 2$	4.8	14.6	22.2	26.2	28.1
$\mathcal{N}_S = 30$; $m = 3$	6.6	18.9	25.1	27.7	29.9

Table 3
Search latency for TTL_Q

Values of TTL_Q	QEAP search latency (ms)
1	190.9
2	380.5
3	572.4
4	762.0
5	951.9

prototype that we would like to deploy on a large virtual test-bed like PlanetLab [35]. We successfully tested our current SPAD prototype on a simple test-bed composed of 4 hosts (each running a super-peer, and several normal-peers) connected via an emulated network provided by another machine running DummyNet [36]. More testing needs to be done before deploying our prototype on a resource-shared test-bed like PlanetLab.

Another limit of our experimental approach is the relatively small numbers of available nodes in the datasets [19,27,28] (these node numbers are presented in Table A1). These numbers do not allow extensive scalability evaluations. A possible approach to overcome this limitation would be to derive a statistical model for Internet end-to-end delay (possibly based on the measurements from these datasets), and combine this model with a topology generator to create large simulation networks. Some contributions study such delay model [37]. Due to time constraints, we did not investigate this approach further.

Finally, one could also argue that the nodes in the studied datasets are not representative of the average real Internet host. Indeed, these dataset nodes are either on corporate or academic networks, and consequently have more bandwidth and less latency than hosts using dial-up connections. However as discussed in Section 1, SPAD is designed for a community of third party service providers competing to propose software elements for *overlay applications*, and cooperating to deliver QoS within these *overlay applications*. As these service providers are business-oriented entities, it is realistic to assume that they would have network connectivity at least similar to the hosts from the studied datasets.

6. Conclusion

This paper presents SPAD, a unique complete architecture that allows the discovery of QoS Enhanced Alternate Paths (QEAPs) on the Internet. The SPAD architecture has the following properties. First, it is a fully distributed system with no single point of failure, where each entity cooperates as an equal peer. More precisely, SPAD is based on an unstructured super-peer system, where cooperative super-peers process QEAP queries issued by their associated normal-peers. These normal-peers receive back incrementally-gathered information on nodes that could potentially act as relays. Second, SPAD is located within a middleware framework, which resides on end-host machines. Therefore, it can be easily and incrementally deployed, as it does not require any change to the current Internet routing mechanisms, or core network components, such as routers inside Autonomous Systems (ASes). Finally, SPAD is efficient in discovering QEAPs on the Internet. Indeed, the experimental evaluations of Section 5 demonstrate that SPAD allows the discovery of a large percentage of the existing QEAPs with acceptable message and latency costs.

Given such properties, and with the use of algebraic characteristics of QoS parameters [8], SPAD is a valuable candidate solution for the provision and management of enhanced QoS between endpoints within a composite service (i.e. an *overlay application*) in the context of service-oriented overlay networks. As described in Section 1, these service-oriented overlay networks would provide complex composite services to mobile network-enabled

devices, thus removing the need for extensive on-board resource provision, and resulting in device cost reduction, lower energy consumption, and enhanced mobility support.

This paper focuses on SPADs design and mechanisms. SPAD uses two complementary schemes to allow QEAPs discovery: (i) a reactive mechanism allowing the exchange of connectivity information among SPAD peers in order to build lists of potential relays, and (ii) a proactive information exchange mechanism improving the construction of these lists. In Sections 4 and 5, we provided detailed descriptions, discussions and evaluations of these two schemes.

To the best of our knowledge, no other contribution provides a fully distributed scheme that allows users at the edge of the network to directly discover and utilize QEAPs. SPAD is designed for a bounded community of peers, such as elementary service providers (Section 1). Compared to the Internet, this community has a relatively small number of hosts. The generalized use of QEAPs, and the deployment of distributed QEAP discovery schemes on all hosts of a vast network like the Internet, would require further studies.

Our current research work focuses on the testing of our SPAD prototype. Once completed, this prototype would allow test-bed and real-world evaluations of the proposed schemes and mechanisms. Finally, we plan to integrate a complete SPAD system within a framework providing generic functions to overlay networks.

Acknowledgements

The authors thank the anonymous referees for their suggestions and comments that have improved the quality of the paper. The authors would also like to thank the RIPE Test and Traffic Measurement project [19], the NLANR Active Measurement project [27], and the PlanetLab All-Pair Ping project [28] for allowing the research community to access and utilize their collections of Internet delay measurements.

Appendix A. Characteristics of QoS enhanced alternate paths

A QoS enhanced alternate path (QEAP) is an alternate Internet path between a pair of hosts. It is built by concatenating a number of end-to-end paths that together offer better end-to-end QoS than the default Internet path, as shown in Fig. 2 of Section 1. QEAPs exist mainly due to the operation of the Border Gateway Protocol (BGP) that provides IP routing between Internet Autonomous Domains (AS) [6]. Routing decisions in BGP are made based on administrative policies and shortest AS hop count. Due to reasons such as economic partnership or competition issues, network administrators may not consider QoS optimization as a primary factor when implementing BGP routing policies. Thus, in some cases, these policies may result in non QoS-optimal default Internet end-to-end paths.

Using trace-based simulations, we analyzed the availability and the characteristics of QEAPs on subsets of the European and American Internet. Table A1 presents this analysis on different measurement datasets from the projects RIPE-TTM [19], NLANR-AMP [27], and PlanetLab All Pair-Ping [28]. These datasets provide snapshots of one-way delay or Round Trip Time (RTT) delay, between hosts within a given community. The analysis accounted for processing delay on potential relay nodes by adding an offset of 1 ms to the delay of all discovered alternate paths before deciding if they were QEAPs. Earlier studies on transport layer processing overhead [38,39] support this offset value.[4] In addition, we removed unusable hosts from the datasets, namely hosts with an invalid or incomplete set of measurements. The results show that: (i) there is a significant number of QEAPs in each of the datasets, and (ii) these QEAPs provide substantial improvements to the end-of-end delay. Figs. A.1 and A.2 present further QEAP analysis on the RIPE-TTM datasets. Around 57% of the QEAPs have a delay constancy of more than 5 min, which is consistent with the results from [26,25]. Fig. A.2 shows, that in more than 90% of the cases, gains that can be made via 3-hop QEAPs are less than gains via 2-hop QEAPs. Other analysis [4] show that most node pairs with existing 3-hop QEAPs also have 2-hop ones. It is clear that the benefit from discovering QEAPs with more than 2-hop is relatively small, thus this paper focuses only on 2-hop QEAP discovery.

[4] SPAD is part of a middleware that mediates applications and networks, similar to the role of the transport layer. Thus, it could functionally be located at a similar level. TCP processing delay is about 1–2 ms/packet when no kernel-to-user-space memory copy is performed, which would be the case within a QEAP relay node.

Table A1
QEAP analysis summary for different datasets

	Dataset		
	RIPE TTM	NLANR AMP	PlanetLab AllPair Ping
Dataset date	25/05/04, 12/07/04	31/01/03, 17/06/04	11/04/05, 27/04/05
Number of valid hosts	47	107	177
Measurement frequency	2/min	1/min	1/15 min
Percentage of pair of nodes with existing QEAPs	50.6%	39.7%	49.5%
Mean delay gain on these existing QEAPs	26.1%	39.7%	16.4%

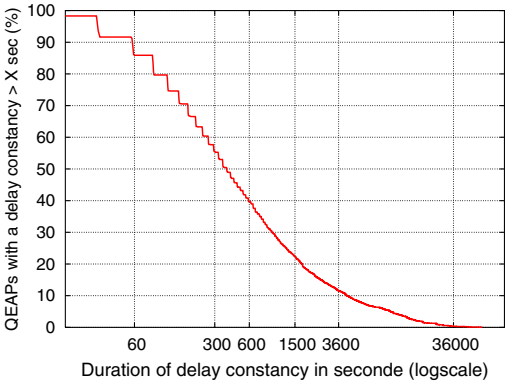

Fig. A.1. Cumulative distribution of the delay constancy duration on QEAPs. RIPE-TTM dataset: 25/05/05.

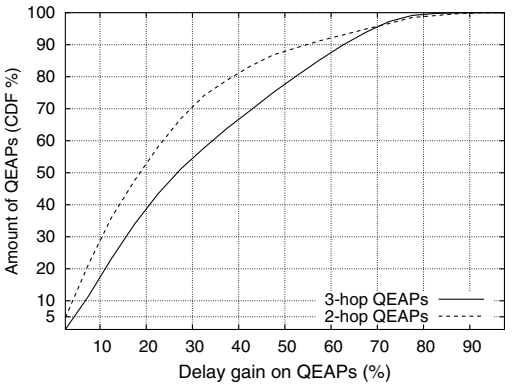

Fig. A.2. Delay gain comparison between 3-hop and 2-hop QEAPs. RIPE-TTM dataset: 25/05/05.

Appendix B. Computation of $M+1$ in a stationary power-law topology

Section 4.3 introduced $M+1$, the average number of direct neighbors for a given node in a graph, and provided a mechanism that allows a node to evaluate $M+1$ in a distributed manner within a dynamic topology. To complete the analysis of Section 4.3.3, this section describes an analytical computation of $M+1$ in a power-law topology that has reached a stationary regime. In such a stationary topology, Barabsi and Albert [40] provide the following expression for the probability $P(k)$ of a node to have k direct neighbors:

$$P(k) = \frac{2m^2}{k^3} \tag{B.1}$$

with m being the minimum node degree of a node. Thus, by definition the expected value $E(k)$ of the average number of direct neighbors for a node is given by:

$$E(k) = \sum_{k=m}^{\infty} kP(k) = \sum_{k=m}^{\infty} k2m^2k^{-3} = 2m^2 \sum_{k=m}^{\infty} \frac{1}{k^2}. \tag{B.2}$$

Since $S = \sum_{k=1}^{\infty} \frac{1}{k^2}$ is the Riemann series, which converges to $\frac{\pi^2}{6}$, we obtain:

$$E(k) = 2m^2 \left(\frac{\pi^2}{6} - \left(\sum_{k=1}^{m-1} \frac{1}{k^2} \right) \right). \tag{B.3}$$

In a stationary power-law topology, the above result (B.3) provides an analytical expression of $M+1$, the average number of direct neighbors for a given node.

References

[1] T. Anderson, L. Peterson, S. Shenker, J. Turner, Overcoming the internet impasse through virtualization, in: IEEE Computer, vol. 38, 2005.

[2] B. Raman, S. Agarwal, Y. Chen, M. Caesar, W. Cui, P. Johansson, K. Lai, T. Lavian, S. Machiraju, Z. M. Mao, G. Porter, T. Roscoe, M. Seshadri, J. Shih, K. Sklower, L. Subramanian, T. Suzuki, S. Zhuang, A. Joseph, R. Katz, I. Stoica, The sahara model for service composition across multiple providers, in: IEEE Pervasive Computing, August 2002.

[3] Skype, <http://www.skype.com>.

[4] T. Rakotoarivelo, P. Senac, A. Seneviratne, M. Diaz, Enhancing qos through alternate path: An end-to-end framework, in: IEEE International Conference on Networking (ICN), April 2005.

[5] T. Rakotoarivelo, P. Senac, A. Seneviratne, M. Diaz, A super-peer based method to discover qos enhanced alternate paths, in: IEEE Asia-Pacific Conference on Communications (APCC), October 2005.
[6] S. Savage, A. Collins, E. Hoffman, J. Snell, T. Anderson, The end-to-end effects of internet path selection, in: Proceedings of ACM SIGCOMM Conference, 1999.
[7] Z. Wang, J. Crowcroft, Quality-of-service routing for supporting multimedia applications, in: IEEE Transactions on Selected Areas in Communications, vol. 14, 1996, pp. 1228–1234.
[8] B. Yang, H. Garcia-Molina, Designing a super-peer network, in: IEEE Conference on Data Engineering, 2003.
[9] D.G. Andersen, H. Balakrishnan, F. Kaashoek, R. Morris, Resilient overlay networks, in: ACM Symposium on Operating Systems Principles (SOSP), 2001.
[10] Z. Li, P. Mohapatra, Qron: Qos-aware routing in overlay networks, in: IEEE Transactions on Selected Areas in Communications, January 2004.
[11] R. Beyah, R. Sivakumar, J. Copeland, Application layer switching: A deployable technique for providing quality of service, in: Proceedings of IEEE GLOBECOM Conference, 2003.
[12] M. Ripeanu, I. Foster, A. Iamnitchi, Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design, in: International Workshop on Peer-to-Peer Systems, March 2002, pp. 85–93.
[13] E. Lua, J. Crowcroft, M. Pias, R. Sharma, S. Lim, A survey and comparison of peer-to-peer overlay network scheme, in: IEEE Communications Survey and Tutorial, March 2004.
[14] T. Rakotoarivelo, P. Senac, A. Seneviratne, M. Diaz, A structured peer-to-peer method to discover qos enhanced alternate paths, in: IEEE International Conference on Information Technology and Applications (ICITA), July 2005.
[15] T. Fei, S. Tao, L. Gao, R. Guérin, How to select a good alternate path in large peer-to-peer systems, in: IEEE INFOCOM, April 2006.
[16] D. Stauffer, A. Aharony, Introduction to Percolation Theory, Taylor and Francis, 1992.
[17] A. Ganesh, A.-M. Kermarrec, L. Massoulie, Peer-to-peer membership management for gossip-based protocols, in: IEEE Transactions on Computers, vol. 52, 2003.
[18] F. Banaei-Kashani, C. Shahabi, Criticality-based analysis and design of unstructured peer-to-peer networks as complex systems, in: Internatiional Symposium on Cluster Computing and the Grid, 2003.
[19] Ripe traffic test measurement, in: http://www.ripe.net/test-traffic/.
[20] R. Braden, D. Clark, S. Shenker, Integrated services in the internet architecture: an overview, IETF RFC 1633.
[21] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, W. Weiss, An architecture for differentiated services, IETF RFC 2475.
[22] Y. Pyun, D. Reeves, Constructing a balanced, (log(n)/loglog(n))-diameter super-peer topology for scalable p2p systems, in: International Conference on Peer-to-Peer Computing (P2P), 2004.
[23] K. Gummadi, S. Saroiu, S. Gribble, King: Estimating latency between arbitrary internet end hosts, in: Proceedings of the SIGCOMM Internet Measurement Workshop (IMW 2002), Marseille, France, 2002.
[24] M. Allman, W. Eddy, S. Ostermann, Estimating loss rates with tcp, ACM Performance Evaluation Review 31 (3) (2003) 12–24.
[25] Y. Zhang, N. Duffield, V. Paxson, S. Shenker, On the constancy of internet path properties, in: ACM SIGCOMM Internet Measurement Workshop, 2001.
[26] H.Rahul, M. Kasbekar, R. Sitaraman, A. Berger., Towards realizing the performance and availability of a global overlay network, in: Passive and Active Measurement Conference (PAM), 2006.
[27] Nlanr active measurement project, in: http://watt.nlanr.net.
[28] Planetlab all-pair ping project, in: http://pdos.csail.mit.edu/~strib/pl_app/.
[29] Z. M. Mao, D. Johnson, J. Rexford, J. Wang, R. Katz, Scalable and accurate identification of as-level forwarding paths, in: IEEE INFOCOM, March 2004.
[30] Ripe routing information service, in: http://www.ripe.net/projects/ris/.
[31] I. Stoica, R. Morris, D. Karger, F. Kaashoek, H. Balakrishnan, Chord: A scalable peer-to-peer lookup service for internet applications, in: Proceedings of the 2001 ACM SIGCOMM Conference, 2001, pp. 149–160.
[32] M. Ripeanu, Peer-to-peer architecture case study: Gnutella network, in: IEEE Conference on Peer-to-peer Computing (P2P), 2001.
[33] Brite topology generator, <http://www.cs.bu.edu/brite>.
[34] Q. Zhang, F. Yang, W. Zhu, Y. Zhang, A construction of locality-aware overlay network: moverlay and its performance, in: IEEE Booktitle on Selected Areas in Communications, vol. 22, 2004.
[35] Planetlab project, <http://www.planet-lab.org/>.
[36] L. Rizzo, Dummynet: a simple approach to the evaluation of network protocols, in: ACM Computer Communication Review, 1997.
[37] C. Bovy, H. Mertodimedjo, G. Hooghiemstra, H. Uijterwaal, P.V. Mieghem, Analysis of end-to-end delay measurements in internet, in: Passive and Active Measurement (PAM), 2002, pp. 26–33.
[38] D. Clark, V. Jacobson, J. Romkey, H. Salwen, An analysis of tcp processing overhead, IEEE Communication Magazine 27 (6) (1989).
[39] S. Makineni, R. Iyer, Measurement-based analysis of tcp/ip processing requirements, in: Annual International Conference on High Performance Computing (HiPC 2003), 2003.
[40] A. Barabasi, R. Albert, Emergence of scaling in random networks, Science 286 (1999) 509–512.

Thierry Rakotoarivelo received his master's degree in Computer Networks from the Institut National des Sciences Appliquees of Toulouse (INSAT, France) in 2001. From 2001 to 2003, he worked as a research engineer at the Motorola Australia Research Center (MARC, now closed), where he contributed to several projects and patent applications within the Sydney Networks Research Laboratory (SNRL), and the Wireless Technology Laboratory (WTL). In 2007, he received his Ph.D. in Electrical Engineering and Telecommunications from the University of New South Wales (UNSW, Australia) in

cotutelle with the Institut National Polytechnique of Toulouse (INPT, France). He then joined National ICT Australia (NICTA), where his current research interests are in peer-to-peer systems, Quality of Service, and overlay networks.

Patrick Sénac graduated from the Ecole Nationale SupÄrieure d'IngÄnieurs d'Hydraulique d'Electrotechnique, d'Electronique et d'Informatique et TÄlÄcommunication (ENSEEIHT) in 1983 and received the Ph.D. degree in computer science in 1996 from Toulouse University, France. He is professor of computer science and head of the Mathematics and Computer Science Department at the Ecole Nationale SupÄrieure d'IngÄnieurs de Constructions AÄronautiques (ENSICA) in Toulouse, France. He is also a member of the research group OLC (Tools and Software for Communication) of the LAAS-CNRS. He has been working during the last decade on the modeling and design of synchronization constraints in distributed multimedia/hypermedia systems. He has published more than 80 papers in international conferences and reviews, has co-edited the proceedings of several international conferences and is the co-author of two books on Petri Nets, one book on multimedia systems and one on pervasive networking. He is involved in several national, European and international research projects on advanced communication architectures. Patrick Sénac is currently leading the French national research group on Communication Networks of the CNRS. During 2004 he was invited professor at the School of Electrical Engineering and Telecommunication of the University of New South Wales in Sydney, Australia. His current research interests focus on the modeling and design of advanced transport protocols and end-to-end communication mechanisms over the Internet.

Aruna Seneviratne is the Director of the Australian Technology Park Laborartory of the National ICT Australia (NICTA). He also holds the Mahanakorn Chair of Telecommunication in the School of Electrical Engineering and Telecommunication at the University of New South Wales (UNSW). He has a Ph.D. from the University of Bath, UK, and a B.Sc (Hons) degree from Middlesex Polytechnic. Since graduating he has held academic appointments at the University Bradford (UK), Curtin University, Australian Defense Force Academy (UNSW) and the University of Technology, Sydney. Outside academia he has worked at the Standard Telecommunication Laboratories (UK), Muirhead and Co (UK), and Telecom Australia (Telstra). He has also spent time at the MASI Laboratory at the University of Pierre Marie Curie (Paris), and the Rodeo Group at INRIA (Nice) as a visiting professor. His current research interests are in mobile data communication systems.

Michel Diaz is Director de Research at the CNRS, Centre National de la Recherche Scientifique, leads the Research Department "Critical Computer Systems" at LAAS-CNRS, and is working on the development of formal methodologies, techniques and tools for designing distributed systems. In 1989 and 1990, he spent a year as a visiting staff at the University of Delaware at Newark and at the University of California at Berkeley. He was the manager of the EC ESPRIT SEDOS project on the development of formal description techniques. He headed the CNET-CNRS project CESAME on the formal design of high speed multimedia cooperative systems and the French TOPASE project on Distributed Multimedia Professional teaching. He was co-ordinating in the EC programme IST the European project GCAP on new architecture for active multicast multimedia end-to-end protocols for the internet. He was Director of the French Research on "Architecture, Networks, Systems and Parallelism" from 1994 to 1999. Since that time he is in charge for the CNRS of co-ordinating the French research groups of Experts on Communication Networks. He is expert for many European and French Programmes. He was member of the Advisory Board on the future of the Internet at the EC and wrote a report for the future of the French national research computer network Renater. He served as Chairman or member of many Program Committees. He has written more than 200 technical publications, one book, and is the editor or co-editor of 11 North Holland, Springer World Scientific, ARAGO and Hermes volumes. He has received the Silver Core of the IFIP, is Senior Member of the IEEE, is member of the New York Academy of Sciences and is listed in the Who's Who in Science and Engineering.