A Super-Peer based Method to Discover QoS Enhanced Alternate Paths

Thierry Rakotoarivelo a,b,c, Patrick Senac b,c, Aruna Seneviratne d, Michel Diaz c

a University of New South Wales, Sydney, Australia
thierry@mobqos.ee.unsw.edu.au
b ENSICA, Toulouse, France
{senac, diaz}@laas.fr
c LAAS-CNRS, Toulouse, France
d National ICT Australia (NICTA), Sydney, Australia
aruna.seneviratne@nicta.com.au

Abstract - In the next generation Internet, the network should not only be considered as a communication medium, but also as an endless source of services available to the end-systems. These services (i.e. Overlay Applications) would be composed of multiple cooperative distributed software elements that dynamically build an ad hoc communication mesh (i.e. an Overlay Association). In a previous contribution, we proposed a collaborative distributed method to provide enhanced QoS between end-points within an overlay association. It aimed at discovering and utilizing composite alternate end-to-end paths that experience better QoS than the path given by the default IP routing mechanisms. Extending this work, this paper presents our investigations on an unstructured Super-Peer network-based method to discover such QoS enhanced alternate path.

I. INTRODUCTION

In the last few years, there has been a steady increasing demand for mobile network-enabled devices. These devices collectively form a pervasive networking environment around the user, informing her, supporting her communication needs, and performing various tasks on her behalf. In that regards, these devices would no longer need extensive local applications, computing or storage resources. Service Providers at the edge of the network would provide them with these now distributed resources. The network would then be an endless source of services available to the end-systems, which would ultimately act only as input/output devices. These services would be composed of multiple cooperative distributed software elements, performing elementary tasks and inter-communicating [1], hence dynamically deploying an overlay network above the existing network infrastructures. We use the term “Overlay Application” to refer to the distributed application composed by such elements. Within an overlay application, data flows no longer travel between just two end-points, but may instead traverse multiple peer end-points (hosting processing application elements) and be produced/consumed by other multiple peers. This defines a new peer-to-peer communication scheme different from the traditional point-to-point, or point-to-multipoint ones. For example, an audio flow in a distributed Voice Over-IP service [2] might pass through several peers providing elementary services (e.g. firewall traversal, codec/quality adaptation). We introduced the concept of Overlay Association: a unique entity referring to the set of connections between elements within an Overlay Application [3]. Using this concept, we could design mechanisms to improve and manage as a whole the overall Quality of Service (QoS) experienced by the end-user(s) of an overlay application (e.g. reducing latency in the above VoIP example). Such mechanisms would perform global resource optimization on an overlay association, as opposed to non-optimal compositions of local resource management decisions. We also introduced Transport Layer Framework [4] that would implement such mechanisms; hence providing unified means to manage communication needs of overlay applications, and reduce their implementation complexity.

In [3], we proposed a distributed peer-to-peer method to enhance the QoS between two peers within an overlay association. This method was deployed on cooperative end-points at the transport level within our overlay framework. It was based on the discovery and utilization of QoS Enhanced Alternate Paths (QEAPs) experiencing better QoS than the default path given by the IP routing mechanisms [5]. This method could then be used in conjunction with algebraic properties (e.g. additive, multiplicative, concave) and composition rules (e.g. association of additive and concave) of QoS metrics (discussed in [6]) to guarantee enhanced QoS to an entire overlay association. We evaluated and validated our method for the one-way delay (additive QoS metric).

In this paper, we address one limitation of this previous work (see section III.A). We propose an extended method where cooperative end-points belong to an unstructured Super-Peer network [7]. Within that network, normal peers
We confirmed their existence on a subset of the European policies between Internet Autonomous Domains (AS) [5].

A. Motivations

Peer-to-Peer (P2P) approaches are based on the unstructured model; we discuss this in section III, and evaluate its performance in section IV. Section V concludes this paper with some directions on our future works.

II. RELATED WORKS

Savage et al. discussed the existence and benefits (enhanced or controlled QoS and robustness) of alternate Internet paths in [5]. Following contributions [8, 9, 10] proposed methods and architectures to discover and utilize such paths. These frameworks all involved either a central entity or several distributed third-party brokers that “sell” QEAPs to end-users. On the contrary, SPAD and our previous method [3] are fully distributed schemes and allow end-users to discover and utilize QEAPs without having to pay any third-party providers.

Within such distributed QEAP discovery methods, nodes play both roles of clients and servers, hence the choice of a Peer-to-Peer (P2P) approach in SPAD’s design. The P2P paradigm has become the focus of several research works and the base of many popular applications (e.g. Gnutella, Skype) within the past few years. In [11], the authors provide a comprehensive overview of these contributions, they study and classify P2P systems into two models: unstructured and structured. They conclude that the best-suited model for a given application depends on its required functionalities.

SPAD is based on the unstructured model; we discuss this design choice in more details in section III.C. Among the multiple contributions in the unstructured area, some focus on traffic pattern analysis [12], or scalability improvement techniques (e.g. super-peer layer, small-world network organization, protocol improvements) [7,13]. To the best of our knowledge, prior to SPAD no other contributions present an unstructured P2P approach to enhance QEAPs discovery.

III. AN SUPER-PEER BASED APPROACH TO ALTERNATE PATH DISCOVERY

A. Motivations

QEAPs exist mainly due to non-QoS optimal BGP routing policies between Internet Autonomous Domains (AS) [5]. We confirmed their existence on a subset of the European Internet and concluded that they provide significant benefits to traffic between two end-points [3]. We proposed a simple method to discover them in a distributed manner within a community of cooperative end-points. That method aims at minimizing a particular QoS parameter (e.g. delay) on each leg of a 2-hop alternate path. Each Transport Entity TE Ni of a node Ni in the community has a small fixed-size list of known candidate relay nodes (L Ni=|TE Ni|), i.e. the known nodes Ni with the shortest delay TE Ni to TE Ni. When searching for a QEAP to node B, node A asks its TE A to send parallel probe messages to potential relay nodes within L A. If no QEAP is found, then TE A retrieves L B from TE B, and sends parallel probe messages to nodes within L B.

One limitation of this scheme is the fact that TE A knows only delay information of the type TE A to TE X and TE A to TE Y. It has only a partial knowledge of the connectivity parameters within the community. Since measurement methods could be asymmetric (e.g. King [14]), there might exist nodes Ni in the community that have delay information of the type TE Nk to TE A or TE NK to TE B. If TE A had access to this information it could send probe message to these nodes and improve its chance of discovering a QEAP (assuming symmetry of the connectivity parameter, see section IV).

The motivation behind the proposed SPAD scheme is to have this information stored in a simple distributed database made of super-peer nodes. Each node Xi in the community associates with a super-node that would keep a copy of its list containing information of type: TE Nk to TE X. When a node requests a QEAP, these super-peer would cooperate to provide the requesting node with relevant information concerning potential relay nodes for that QEAP.

B. SPAD Description

During its initialization phase, a transport entity TE A contacts a set of super-peer nodes SP i. We assume the existence of bootstrap mechanisms to provide new nodes with super-peer contact details, and to elect new super-peers among normal peers when needed (as done in existing P2P applications). TE A requests would provide TE A with small lists of random nodes already part of the community. TE A would evaluate its delay to some of these random nodes, and would keep in a list L A the ones with the lowest delay. Thus L A contains entries of the form <TE A, TE X, delayAtoX>. We note that if A and X are on the same AS (myAS), they would probably share the same gateway towards B (another node not on myAS), thus the paths AtoB and XtoB would be similar, and the chances that a QEAP exists from A to B via X would be low. Therefore, we use the following simple mechanism to ensure node diversity in a given list L A. The RIPE RIS project gathers and compiles BGP routing information 3-times a day from over 300 routers worldwide [15]. Based on those data, it provides an IP-to-AS mapping service to the Internet community via a dedicated “riswhois” server. When joining the SPAD community, a node X queries that server to determine its AS number. During TE A’s initial delay evaluation (described above), potential X A, send
their AS number to A. TE_A uses that information to filter out  
X's on the same AS as itself, and X's on redundant ASes (e.g.  
if X_1, X_2, X_3 have the same AS number, TE_A will consider  
only one of them for inclusion in its L_A). Since each node in  
the SPAD community makes a unique “riswhois” query only  
onece, the associated load should not significantly impact the  
normal working load of a server designed to provide public  
services to the Internet community.

After building its list L_A, TE_A randomly selects one super-  
peer S (on a different AS than A, and among the initial  
bootstrap SP's) to associate with, and upload L_A to it. As TE_A  
discovers other nodes, by application connection request,  
it will update its list accordingly. TE_A will also periodically  
re-evaluate its delay towards the nodes X_i to keep L_A  
accurate. Existing work on Internet delay constancy [16]  
could be used to select appropriate re-evaluation intervals.  
TE_A will forward all these updates to TE_S. TE_S maintains a  
collection of lists \{L_{N_i}\} received from its set of associated  
nodes \{N_i\}. To keep a manageable load, super-peers have a  
maximum number k of associated peers, i.e. maximum size  
of a cluster of peers for a single super-peer.

Upon receiving an application connection request toward  
another node B, TE_A evaluates the delay on the default  
Internet path from A to B, and constructs a request of the  
type <requestID, src, dst, delay, TTL>, with src being A's  
address, dst the address of B, and TTL the time-to-live for  
that request. TE_A passes that request to its super-peer S,  
which executes the query-processing algorithm (fig. 1).  
Within this algorithm, a super-peer selects all entries within  
its collection \{L_N\} that match the request, in order to build  
its part of the response (line 6-11). Then if the request’s TTL  
> 0, it forwards the request to all of its connected super-  
peers, and wait for their responses (line 12-17). Upon  
receiving them, it aggregates them with its own (line 15). It  
then executes a filtering process (line 18), before sending it  
back to the source of the request (line 19). The filtering  
process (fig. 2) removes redundant (line 3, 5-8) or irrelevant  
entries from the aggregate responses (line 10, 12-15, 16-19).

Following these mechanisms, replies to a request travel back  
along the request path towards TE_S. Super-peers on that path  
successively filter these entries, decreasing the response size,  
and returning a response containing only the relevant entries.  

When the search process is complete, TE_S receives a list  
of entries of the form <TX_E, TX_Y, delayXtoY> (with \{X or  
Y\} = \{A or B\}). TE_A organizes these entries into 4 sub-lists  
corresponding to AtoT, UtoA, BtoV and WtoB. It first analyzes these lists to search for cases where \{T or U\}={V or  
W}, if such cases exist, TE_A sends probe messages to these  
nodes. Otherwise, it sends probe message in parallel to all the  
nodes in one of these sub-lists. Such probe message allows  
TE_A to find out if the path A_X_B is a QEAP (i.e. TE_X  
evaluates its delay to B), if so it is also used to ask TE_X if it  
would like to be a relay in a QEAP from A to B. An  
admission control function on TX_E replies to TE_S (i.e. if TE_X  
accepts to be a relay, it sends delayXtoB to TE_S, otherwise it  
sends an infinite delay). If TE_S fails to discover a QEAP, it  
executes a similar probing procedure on the next sub-list.  
When a QEAP is found, the involved nodes set up the path,  
use it and monitor it in a similar manner as described in [3].

C. Discussion

We note that a structured P2P system could also provide a  
similar distributed database service. For example, when  
using a Chord-based [17] approach, a new node might need  
to upload its list entries at different location within the  
structured community. This may increase initialization  
complexity, but also decrease search cost (as all existing  
information about a node would be located on a single peer).  
This is a similar problem as designing efficient keyword  
search on structured P2P systems. As noted in [11], possible  
solutions have been proposed, but their efficiency (compared  
to unstructured P2P-based solutions) still needs to be proven.  

Theoretical lack of scalability is the major limit of  
unstructured P2P systems based on control query flooding.  
However, the use of a super-peer structure significantly
extends further this limit [11]. Furthermore, several studies [7, 13, 18] present design methods, query-processing algorithms, and topology construction protocols aiming at improving super-peer systems scalability. These techniques and mechanisms could be readily included in SPAD. Finally, other measurement studies suggest that unstructured P2P systems tend to self-organize into power-law topologies, thus scaling more optimistically in regards to query-flooding [12].

There are some other issues that need more investigations in the proposed scheme: such as security and trust concerns in both super-peer query processing and relay node selection, or the admission control procedure on a candidate relay node. Some other issues, more related to our overall overlay transport framework, are also not discussed here such as QoS management for an entire overlay association. We will investigate them in our future works.

IV. SIMULATION RESULTS AND ANALYSIS

Current topology generators do not provide any realistic model of Internet delays [19]. For this reason, we chose trace-based simulation to evaluate SPAD. We used the NLANR-AMP RTT measurements between 107 hosts across the North-American academic Internet [20] (31/01/03 and 17/06/04). These simulations do not account for dynamic change in network connectivity parameters. This is acceptable for a first evaluation of SPAD. We plan to develop a prototype allowing more comprehensive evaluations.

We first present some preliminary results concerning alternate paths on the NLANR-AMP network, briefly comparing them with similar results from our previous study [3] based on a European network [21]. In both cases, we note a strong symmetry correlation for the delay between two end-points. For 10 000 random 3-tuples <src, dst, time>, we computed a correlation coefficient of 0.998783 on the NLANR dataset. This result supports the previous assumption of using delay \( AtoX \) as an approximation for delay \( XtoA \) (section 3.1). Using similar experimental procedures as in [3], we found an average number of existing QEAPs of 26.41% (stdev: 0.3088) on the NLANR dataset. This is significantly less than the 50.6% observed on the European network. This result is expected, as NLANR nodes are all academic nodes connected to the common research-dedicated HPC (High Performance Connection) backbone, where BGP policies are not subject to the same non-QoS optimal decisions as in the European case [21], thus leaving less opportunities for QEAP occurrences. Due to the same BGP-related reasons, the QEAP gain distribution is also different from the results in [3]. On fig. 3, the majority of the gains (i.e. 53.77%, stdev: 2.1546) are below 20%; they were above that value on the Western European network. The outlier spike at 100% is specific to the NLANR dataset. It corresponds to 3-tuples experiencing temporary failure on the default Internet path between src and dst at time. For these pairs, all alternate paths are indeed QEAP with a gain of 100%. Despite these less favorable results, we selected the NLANR dataset to evaluate SPAD, as it provides more than twice as many nodes as the dataset used in [3].

We randomly selected 10 nodes to perform as super-peer peers, and randomly associated the remaining 97 nodes to them. We fixed the maximum cluster size to 10, and the \( L_{Ni} \) size to 6. We used topologies from [22] to connect the super-peers following a Barabási-Albert power-law model. The following results are averaged over 10 000 trials.

Fig. 4 presents a performance comparison between SPAD, our previous 2-Level QEAP Search scheme (P2LS) [3] and a Random Search scheme (RAND) where a node looking for a QEAP probes a fixed-size list of randomly selected potential relay nodes. The Performance axis corresponds to the percentage of existing QEAP that are discovered by each method. Starting with SPAD, we varied the TTL 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 category and we used that value to set the list size for P2LS and RAND. For a TTL>1, the SPAD method significantly outperforms the 2 other methods, discovering 77.58% (stdev: 0.5726) of existing QEAP for a TTL of 4. The performance of SPAD is function of the existing QEAP that are discovered by each method. Starting with SPAD, we varied the TTL 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 category and we used that value to set the list size for P2LS and RAND. For a TTL>1, the SPAD method significantly outperforms the 2 other methods, discovering 77.58% (stdev: 0.5726) of existing QEAP for a TTL of 4. The performance of SPAD is 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. In [7], the authors demonstrate that in a power-law organized super-peer network, the reach is a function of the existing number of super-peers, their average out-degree, and the request’s TTL. They provide an extensive study of the
This gain comes with a limited additional cost in query forwarding and processing between super-peers. This is an acceptable trade-off, since by definition super-peers are altruistic peers with more computing and networking resources than normal peers.

Our future research plans include more investigations on distributed QEAP discovery methods, the implementation of prototypes to evaluate their dynamic behaviors, and the continuation of our study of an overlay transport framework.

V. CONCLUSION

In this paper, we extend our previous work on QoS Enhanced Alternate Path (QEAP) discovery [3]. We present and evaluate SPAD, an unstructured Super-Peer network-based method that allows QEAP discovery in a distributed manner; thus providing enhanced QoS between two endpoints within an overlay association. For similar list sizes, SPAD allows the discovery of more QEAPs than our previous scheme, with a lower cost in probing messages. This gain comes with a limited additional cost in query forwarding and processing between super-peers. This is an acceptable trade-off, since by definition super-peers are altruistic peers with more computing and networking resources than normal peers.

Table I presents the search (i.e. query forwarding) costs in message number and latency between super-peers (not including processing latency). As expected, both costs raise with the TTL. We note that the latency associated with the chance of discovering 77.10% of existing QEAPS is still close to 200ms. In the context of a VoIP call, we believe that it is a reasonable connection set-up time, since the resulting QEAP would probably be used for several minutes. Table I also shows the average size of responses generated by a request for a version of SPAD with response filtering on super-peers (default behavior as described in fig. 2) and a version of SPAD without filtering. We note that using response filtering on super-peers significantly decrease the size of responses sent back to the originator of a request, thus decreasing the associated bandwidth cost.

TABLE I. COMPARISON OF SPAD’S CHARACTERISTICS FOR DIFFERENT TTL VALUES

<table>
<thead>
<tr>
<th>TTL</th>
<th>SPAD Performance (%)</th>
<th>Search Cost (number of messages)</th>
<th>Search Cost Latency (ms)</th>
<th>Average Size of Returned Response List (number of entries)</th>
<th>With Filtering on SuperPeers</th>
<th>Without Filtering</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>58.59</td>
<td>2.85</td>
<td>97.6</td>
<td>8.26</td>
<td>11.18</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>70.33</td>
<td>6.12</td>
<td>160.8</td>
<td>11.98</td>
<td>17.22</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>77.10</td>
<td>9.28</td>
<td>208.4</td>
<td>15.22</td>
<td>22.75</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>77.58</td>
<td>10</td>
<td>222.1</td>
<td>15.87</td>
<td>23.86</td>
<td></td>
</tr>
</tbody>
</table>

The influence of these parameters on the scalability of such networks.

Fig. 5 presents a comparison of the cost of probe message toward candidate relay nodes. SPAD’s technique of dividing the complete response list into 4 sub-lists and probing each list successively results in a lower cost than RAND and P2LS. For an average list size of 16 candidate relay nodes (resulting from a TTL=4), SPAD needs to probe about 14 nodes to discover a QEAP. This gain comes at the price of an increased latency in QEAP discovery compared to a scheme without list division.

Table I presents the search (i.e. query forwarding) costs in message number and latency between super-peers (not including processing latency). As expected, both costs raise with the TTL. We note that the latency associated with the chance of discovering 77.10% of existing QEAPS is still close to 200ms. In the context of a VoIP call, we believe that it is a reasonable connection set-up time, since the resulting QEAP would probably be used for several minutes. Table I also shows the average size of responses generated by a request for a version of SPAD with response filtering on super-peers (default behavior as described in fig. 2) and a version of SPAD without filtering. We note that using response filtering on super-peers significantly decrease the size of responses sent back to the originator of a request, thus decreasing the associated bandwidth cost.

V. CONCLUSION

In this paper, we extend our previous work on QoS Enhanced Alternate Path (QEAP) discovery [3]. We present and evaluate SPAD, an unstructured Super-Peer network-based method that allows QEAP discovery in a distributed manner; thus providing enhanced QoS between two endpoints within an overlay association. For similar list sizes, SPAD allows the discovery of more QEAPs than our previous scheme, with a lower cost in probing messages.