Interdomain Path Computation for PCE-assisted Traffic Engineering

L. Bisti, E. Mingozzi, G. Stea
Dipartimento di Ingegneria dell'Informazione, University of Pisa
Via Diotisalvi 2, 56122 Pisa – Italy
{luca.bisti, e.mingozzi, g.stea}@iet.unipi.it

Abstract—**Interdomain Traffic Engineering (TE) across domains employing Path Computation Elements should allow a source domain to select a good AS path, i.e. one likely to allow the actual setup of an interdomain tunnel. This is impossible if the AS path is computed online during path setup (which happens, e.g., in per-domain tunnel setup). On the other hand, the routes that a source domain learns from BGP are normally too few, and oblivious to TE constraints. In this paper, we present the Inter-AS Path Computation Protocol (IA-PCP), whose purpose is to compute a larger number of "good" AS paths, allowing a source to choose the suitable one among a larger set, so as to route PCE-assisted interdomain path computation. We show that, within a reasonable computation time, sending relatively few messages and preserving domain confidentiality, IA-PCP provides a source domain with a set of AS paths which closely approximates the one found exploiting full knowledge of the network AS-topology.**

Keywords—Interdomain Traffic Engineering; AS path; Path Computation Element

I. INTRODUCTION

Inter-domain Traffic Engineering (TE) has become a hot topic in the literature, as testified by both the amount of related literature in the recent past (see e.g. [1] and the references included therein) and the standardization activity within the IETF ([2]-[8], [14]-[15]). One of the major drivers behind this is to extend the reach of value-added services, such as Virtual Private Networks (VPNs) beyond the borders of a single domain [9]. A major impairment in crossing domain borders with TE tunnels is represented by the intrinsic decentralized model of the Internet, which is composed of *Autonomous Systems* (ASs) or *domains*, whose topologies and resource provisioning should not be disclosed outside domain borders. As TE is actually concerned with computing routes with associated resources (i.e., reservable bandwidth), not knowing the entire network topology and provisioning make this problem particularly difficult.

A well-known standard for Internet TE is Multi-Protocol Label Switching Traffic Engineering (MPLS-TE, [1]). Recent research in the field of MPLS-TE has led to the standardization of the *Path Computation Element* (PCE, [3]-[5]). The latter is a centralized element which can be queried in order to compute constrained Traffic Engineering Label Switched Path (TE-LSP) between two endpoints of the *area* under its control (i.e. AS or subset thereof. For the sake of conciseness, we assume that a single PCE manages a whole AS throughout this paper. Extensions to the general case of multiple PCEs per AS or one PCE managing more than one AS can be easily inferred by the alert reader). A recent proposal [6] addresses the computation of optimal (i.e., shortest) constrained *interdomain* TE-LSPs based on PCE support, through the *Backward Recursive PCE-based Computation* (BRPC). Given a sequence of domains (or *AS path*) to traverse, the last PCE along the path computes one or more path segments within its domain, and passes the results backwards, so that each PCE adds its own segment. The source domain PCE is therefore presented a set of end-to-end paths, among which it can select the shortest one to be setup through the *Resource Reservation Protocol – Traffic Engineering* (RSVP-TE). The proposal in [7], instead, advocates end-to-end path computation to take place online (i.e., during RSVP-TE signaling), in the forward direction: each domain computes its own segment of the path (possibly, though not mandatorily, through its PCE), and contacts the next downstream domain, advancing towards the destination. If a domain is unable to advance the computation (e.g., due to policy mismatch or lack of TE bandwidth), crankback is required in order to try alternative routes. Crankback requires a considerable amount of signaling and can be time consuming, since many attempts may be needed before achieving success.

In both the above proposals, the problem of selecting the sequence of domains is explicitly dismissed, although the latter has a key importance in the interdomain TE process, as different AS paths have very different performance as far as (e.g.) delay or reliability are concerned. It is instead stated that the AS path is either *discovered* (i.e., computed online), or *predetermined*, i.e. computed offline before the PCE-assisted path computation. The first approach presents a major drawback. In fact, it allows the source domain little or no control regarding the ASs that the computed LSP will traverse, as each PCE is allowed to make routing decisions according to its own policies. Exclusive constraints (e.g., "avoid AS x") possibly specified at the source domain are clearly not enough to steer the AS path search in a controllable way. On the other hand, computing the AS path offline gives full control to the source domain, which is advocated in recent works (see e.g. [27]), and discussed as a viable option within the PCE charter itself [17]. However, there is no means, as of today, for a source domain to precompute *TE-feasible* AS paths for a given TE-LSP request. We say that an AS path is TE-feasible if: i) it is such from a business and regulatory point of view (i.e., policies at involved ASs will allow the setup of the TE-LSP along that AS path), and ii) can be verified, at least at a coarse level, to possess enough TE-resources to allow the transit of the required TE-LSP, so that the subsequent RSVP-

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7262
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7262

TE session through it has a reasonable chance of success. In fact, in today's Internet the AS paths to a destination known at a source domain are those advertised by BGP, which are unlikely to be TE-feasible: on one hand, BGP peering does not necessarily imply willingness to accept interdomain TE-LSP; on the other hand, BGP AS paths are oblivious to TE constraints, so that a BRPC or an RSVP-TE session along a BGP-inferred AS path has no guarantee to succeed. Furthermore, BGP routes are *few*, being no more than one per neighboring AS to a destination (although extensions to BGP to allow several paths per destination are currently being discussed within the IETF, [22]).

A preferable solution would be to present the source domain with a possibly wide selection of precomputed TE-feasible AS paths to a destination, allowing it to choose among them. Besides allowing choice to the source domain, this would also minimize the occurrence of crankback in per-domain path computation, or of failures of BRPC. Depending on the algorithms employed by the PCEs to compute per-domain paths, the time overhead of crankback and BRPC failures can be non negligible. Furthermore, it would enable a source domain to select AS-level diverse paths to the same destination if protection is required. A protocol for computing AS paths should be *effective*, i.e. able to capture a good summary of TE-feasible AS paths, capitalizing on the Internet path diversity while at the same time being selective enough to filter out irrelevant information. For instance, if a source domain is looking for short AS paths to a destination which is known to be three ASs away, it might also be interested in knowing a few suboptimal AS paths of four hops; however, as the number of hops grows, both the relevance of the paths decreases and their number increases, making the effort of computing them all as burdensome as it is pointless. Furthermore, it should be *scalable*, i.e. amenable to large networks and able to carry out its computations in reasonable time ("reasonable" being measured in the timescale of TE practices, where tens of seconds are not a problem), without neither consuming an excessive bandwidth nor overly burdening the servers. Last, but certainly not least, it must not require domains to expose their own internal structure or resource provisioning, so as to preserve confidentiality [15].

The contribution of this paper is the description and evaluation of the *Inter-AS Path Computation Protocol* (IA-PCP), which has been designed having in mind the above requirements. The IA-PCP has its roots in the TE framework developed starting from 2006 within the IST-EuQoS project [10]-[11], [18], [33]. Although both [18] and [33] mention the IA-PCP, neither describe the protocol nor assess its performance via simulation. The protocol runs among per-domain servers. On receipt of an AS path computation request, each server i) verifies source-specified constraints on the number of hops, the minimum bandwidth, QoS metrics; ii) selectively forwards the requests to its downstream neighbors, and iii) collects their responses and assembles a reply to its own upstream neighbors. However, in doing this, each server can *filter* the received set of partial AS paths, so that only the *best* N are advertised. In principle, each domain can rank the received AS paths according to domain-specific policies. However, we show in this paper that, if a coherent *AS path ranking function* is employed by all the server (e.g., minimum number of hops, better QoS, etc.) the results are nearly optimal, meaning that the set of returned AS paths closely approximates the one provided by a reference link-state computation (at the AS level) performed by the source AS.

The rest of the paper is organized as follows: Section II provides a background on the PCE and the existing solutions for inter-domain path computation. IA-PCP is described in Section III. Section IV reports simulation results. We briefly describe the EuQoS path computation framework, which includes IA-PCP, in Section V, and draw conclusions in Section VI.

II. Background and Related Work

This section briefly introduces the PCE and the BRPC procedure. It then puts our contribution into perspective by reviewing some related work on inter-AS path computation.

The rationale behind delegating path computation to a centralized server, i.e. the PCE, rather than having it done online by Label Switch Routers (LSR), is that the latter is a CPU-intensive task. Moreover, domain policies which may constrain path computation can be managed more easily with a centralized server. A first series of RFCs ([3]-[5]) describe the architecture, the requirements for the PCE communication protocol and the requirements for the discovery of PCE. The architecture is composed of three main functions: i) The PCE itself which is in charge of computing the path; ii) a Path Computation Client (PCC) which queries the PCE for a path computation, and iii) a PCE Communication Protocol (PCECP) which implements the communication between PCEs and between the PCC and the PCE. The result of the PCE computation is an Explicit Route Object (ERO), which the head LSR itself uses to setup the TE-LSP.

When the two ends of an LSP are under the control of different PCEs, the PCEs managing the various ASs/areas in the interdomain path cooperate to compute the whole path. The *Backward Recursive PCE-based Computation* (BRPC, [6]) can be used once all the PCEs in the path have been located (e.g., if the AS path has been previously computed). In the latter, the computation starts at the tail-end PCE, which returns *all* the possible paths to the destination coming from any peering point in the AS managed by the upstream PCE. The upstream PCE, in turn, adds all the possible paths from an ingress to one of the selected egresses, and so on. The source domain is therefore presented a list of partial, per domain sub-paths, which it can arrange into an optimal (i.e., shortest) end-to-end path by applying the well-known Dijkstra algorithm. Note that online per-domain path computation in the forward direction cannot achieve global optimality, even if each AS computes locally shortest sub-paths.

A number of papers have appeared in the recent past related to interdomain TE in a PCE-enabled network. Those concerned with *offline* AS path computation generally require each AS to export a summary of their topology and TE characteristics, e.g. a set of ingress-egress TE tunnels associated with the available TE capacity and possibly other metrics (e.g., QoS, monetary cost). In [19], it is assumed that each PCE possesses the *Aggregated Representation* of the whole set of ASs in the network, including the available disjoint paths that connect each AS border nodes. Based on the latter, a source PCE can compute minimum cost paths to any destination. Means for distributing the required information are

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7262
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7262

however not investigated. A similar solution is proposed in [20]-[21], where a *centralized* repository is envisaged, which maintains an aggregate representation of all the involved ASs, and can be queried from a source AS in order to compute AS paths. Both the above approaches have several drawbacks. First, and foremost, that of poor scalability: both solutions are in fact explicitly designed for small-scale networks, involving a limited set of ASs. Exporting aggregated representations of all the ASs, keeping their TE information up to date, and computing the path in a centralized way (at a source PCE, or, worse yet, at a centralized server), are likely to become unmanageable as the network size increases: it is shown in [21] that path computation with N ASs is expected to be $O(N^4)$. Second, they require a high degree of mutual trust among the participants. In fact, exposing aggregated topologies, TE capacities, QoS capabilities, and even transit costs [21], which is required in the decision algorithms, may not meet wide acceptance in a less-than-fully cooperative environment. Third, having a centralized repository does not allow single ASs to deploy complex and individual policies as for costs and transit permissions: for instance, the solution in [21] implicitly assumes that bandwidth is charged linearly per unit by all ASs, and that exporting a single transit ingress-to-egress tunnel implicitly allows every remote AS to set up an LSP through it. This is clearly not representative of the complexity of today's inter-AS business policies. PCE-assisted interdomain path computation is also considered in [28]. Authors propose to enhance BGP so as to carry QoS information and to separate the routing planes for different classes of service. However, it is assumed that the AS path is computed by the source domain PCE, relying on the information carried by the enhanced BGP.

Interdomain TE solutions adopting per-domain computation rely on *online* path computation (although, as already observed, they might still benefit from precomputed AS paths). In that respect, it is often assumed that the next downstream AS is located by looking at BGP tables. Once this is accomplished, the related egress point is located and the intradomain part of the path is computed. When more than one BGP route exists, [25] proposes to select the egress point through delay-proportional IGP metrics or Vivaldi coordinates [26]. Recently, extensions to crankback signaling have been proposed to allow RSVP-TE to search for AS paths having minimum length [23]. As observed in [24], shorter AS paths are not necessarily associated to better QoS. Moreover, a source domain may also want to trade the AS path length for a larger minimum available TE capacity, or for higher reliability. Finally, recent solutions for interdomain routing in the Internet, though not devised for TE purposes, appear to stem from similar motivations as ours, and are thus worth mentioning. For instance, MIRO [13] proposes that a source domain directly contact - using an ad hoc protocol - neighboring or remote ASs to learn partial or full AS paths to a destination when not satisfied with those learned from BGP. NIRA [27] develops a topology information propagation protocol for a user to discover partial AS paths *excluding* the core of the Internet. The user then assembles partial AS paths into an end-to-end route. Obviously enough, the AS path search has no TE-feasibility constraints in both cases.

III. The Inter-AS Path Computation Protocol

This section describes the Inter-AS Path Computation Protocol (IA-PCP). We first settle down the architectural assumptions underlying the protocol. We then introduce its functionalities and describe its properties at the end of the section.

A. Architectural Assumptions

We assume that, in each AS, one entity exists which possesses knowledge of i) the negotiated Service Level Agreements with its neighbors, therein including support for interdomain TE and TE capacity limits; ii) *all* the BGP routes advertised by BGP neighbors; iii) the available TE capacities and the QoS (e.g., the delay) on edge-to-edge paths. The IA-PCP is run by an entity possessing the above characteristics. While it is not in the scope of this paper to define such an entity, the following comments are in order:

- An entity managing some of the domain administrator information base in order to automate provisioning decisions has been envisaged in many recent QoS-oriented architectures, e.g. in ITU-T Next Generation Network [12], or in the IST-EuQoS and IST-MESCAL projects [11], [28]. The EuQoS *Traffic Engineering and Resource Optimization* (*TERO*) module actually possesses all the above requirements.
- The imperfect or partial knowledge of some of the above requirements hampers the *effectiveness* of IA-PCP, reducing the likelihood that the computed AS paths are TE-feasible, but not its correctness. Similarly, taking into account a larger information base (e.g., interdomain traffic matrices, resource provisioning at the various queues, etc.) may improve the quality of the computed AS paths.
- The PCE does not possess all the above information.

Our solution explicitly avoids to make reference to a specific interdomain TE business model (such as the one considered in [20]-[21]). Instead, we allow ASs the maximum freedom in negotiating bi-lateral (or even multilateral) SLAs related to interdomain TE, as long as this translates to an IA-PCP agent knowing the maximum TE capacity it can reserve along an interdomain link to a neighboring AS for a destination and a QoS characterization of the same. For the sake of presentation, we assume that BGP reachability of a destination through a neighboring AS is a *necessary* condition for interdomain TE reachability through the same path: in fact, it seems reasonable that, if a provider is not going to forward IP packets from a source towards a destination, it will hardly be willing to pin TE resources for the same source/destination pair. This, however, does not exclude other options, e.g. interdomain TE reachability being advertised explicitly through BGP itself, without affecting our proposal. Such issues are however outside the scope of this paper.

Finally, the IA-PCP protocol is meant to run at the *application* level, assuming in-order reliable delivery. For instance, the EuQoS implementation uses XML encoding over HTTP. Furthermore, local server reliability mechanisms are assumed, so as to avoid that a server that crashes loses the state related to ongoing AS path computations.

B. Protocol Description

A path computation process is started by an LSP request to the IA-PCP agent managing the source AS. The request is

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7262
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7262

characterized by the source and destination AS numbers, a maximum number of AS hops that the request is allowed to traverse, a minimum required bandwidth, and constraints on end-to-end QoS parameters (delay, jitter and loss rate) [31].

The IA-PCP protocol consists of three phases, namely: 1) Path Discovery; 2) Path Collection; 3) Path Commit/Cancel. Each phase requires the exchange of protocol messages between IA-PCP agents managing neighboring ASs. Five types of messages are defined to implement the various protocol tasks, i.e. *Request*, *Response*, *Error*, *Commit* and *Cancel*, whose function will be detailed in the remainder of this section. Their format is shown in TABLE 1. We first describe the three phases without taking into account reliability issues (such as the possible occurrence of deadlocks), and then describe how to handle the latter using timeouts. Furthermore, we observe that the current implementation of IA-PCP allows several AS path computations to take place in parallel. For simplicity of exposition, a single path computation is assumed to be in place at each node at a time. Finally, although we describe the protocol aiming for generality, we will sometimes refer to the scenario represented in Figure 1, where each node represents an AS, and the path computation is from AS 1 to AS 4. BGP routing table entries related to AS 4 are also reported above each node. In order to present a complete scenario, we also assume that, in the example, all AS paths between AS 1 and AS 4 are TE-feasible and meet all QoS requirements. A sequence diagram is also reported in the figure, which will be commented throughout the paper as the relevant concepts are introduced.

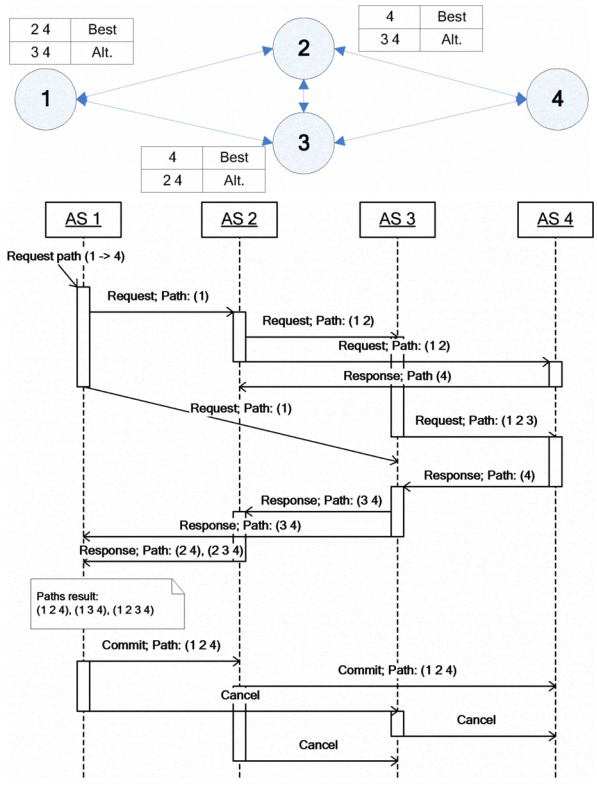

Figure 1. Reference scenario and sequence diagram

The IA-PCP agent of the source triggers a new path computation process. As a first step, it generates a unique *Path Computation Identifier* (*PCID*), which univocally identifies a specific IA-PCP computation. The PCID indexes the relevant protocol status information maintained by an IA-PCP agent, which allows several independent path computations to be processed concurrently at a server. A PCID can e.g. be computed by composing the source AS number and a timestamp.

Path Discovery: The request from AS1 starts the Path Discovery phase, during which the latter is propagated through neighboring IA-PCP agents according to the protocol rules, until either a constraint is violated or the IA-PCP agent of the destination AS is reached.

TABLE 1. IA-PCP MESSAGE FIELDS

All messages	ID Object: {PCID, PRID, Type, Source AS, Dest AS, Dest Network Prefix, RankBy, HopCount}
Request message	Partial AS-Path from Source AS (PATH), Residual QoS budget, Requested bandwidth
Error message	Error Code
Response message	AS-Path Info Object #1: {PATH, BW, CumQoS}, ..., AS-Path Info Object #N

The *ID Object* field (which is common to all messages) includes several subfields: first of all, the endpoints for the LSP, i.e. the *Source* and *Destination* AS numbers, and the *Dest Network Prefix*, i.e. the IP address of the destination network within *Dest AS*. Furthermore, it reports both the PCID generated by the source agent and a Path Reference ID (PRID), having the same structure as the PCID but with a *local* meaning. In particular, it identifies the AS which *sends* the Request message (whereas PCID refers to the *source* AS). The PRID is used to identify sessions between neighboring ASs in a path computation, which allows an agent to keep track of which messages it received from (or sent to) whom. The *Type* field identifies the message as being a Request. The *RankBy* field denotes the preferred *AS path ranking function* at the source. The latter is the function according to which the ASs involved in the path computation should rank the AS paths. For instance, a source may be looking for *short* paths, including as few AS hops as possible, or, instead, for paths with *low delay*. More details on the set of possible functions are given later on, after the Path Collection phase has been described. The *hopCount* field represents the maximum number of ASs that the Request message is allowed to traverse. The *Partial AS-Path* is the sequence of ASs traversed so far by the Request message. When a domain forwards a request, it appends its own AS number to the list. The *Residual QoS budget* represents the QoS constraints for the *remainder* of the AS path, i.e. from the AS receiving the message down to the destination. The *Requested bandwidth* field specifies the minimum bandwidth constraint. In the example, the Request message is forwarded by AS 1 to all its neighbors advertising routes to AS 4, i.e. AS 2 and AS 3.

On receipt of a Request message carrying a *new* PCID, each IA-PCP agent stores it in a local database and decreases the hopCount field. If the latter is null and this is not the destination AS, it terminates the path discovery and sends an Error message as specified further on in the paper. Otherwise, it computes a set of TE-feasible next-hop ASs, to which the Request is to be forwarded, as follows:

1. It computes $H = H_{BGP} \cap H_{SLA}$, where: H_{BGP} is the set of BGP next-hop ASes for the destination, obtained by considering *both* the best and the alternative BGP routes; H_{SLA} is the set of the neighboring ASs for which an SLA has

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7262
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7262

been negotiated that allows the transit of inter-domain LSPs with the specified characteristics. Note that the agent can obtain H_{BGP} by considering both the *Dest AS* and the *Dest Network Prefix*. This preserves the effectiveness in the presence of BGP route summarization.

2. It purges H by removing any AS already included in the partial AS-path, so as to avoid creating loops.
3. For each AS $i \in H$, it reads from a local database Q_i and B_i, i.e. the *QoS parameters* (Q) and *TE bandwidth* (B) associated to the local path segment spanning between the ingress points of the local domain and AS i. If more than one such path exists (e.g., because several ingress-egress couples exist or, more likely, because several intradomain paths are available), the IA-PCP agent may select one at will, or perform *maximum* (*minimum*) computations on the delay (TE bandwidth).
4. For each AS $i \in H$, the bandwidth and QoS constraints are checked. If $BW_{LSP} > B_i$, or if $Q_{REQ} < Q_i$, then AS i is removed from H.
5. If $|H| > REQ_{MAX}$, where the latter is a local preconfigured value, select a subset of H having REQ_{MAX} elements according to local policies. Such policies might be inferred, for instance, from the objective of the path computation as specified by the source (e.g., select those next-hops through which shorter routes to the destination exist), or they may be related to *local* provisioning policies (e.g., select the less congested intradomain paths), or both. This last clause allows large (backbone) ASs, having a very large degree of connectivity, to limit the amount of traffic processed at their IA-PCP agent.

A new *outgoing* Request is then assembled as follows:

- The *ID Object* is replicated. The *hopCount*, is decreased, and the *PRID* is overridden with the one generated using the local AS number.
- The local AS number is appended to the *Partial AS-Path*.
- The *Requested bandwidth* is copied.
- The *Residual QoS budget* is updated by deducting the AS contribution from the value in the received Request, according to the appropriate QoS concatenation function [31] (e.g., a simple subtraction for the delay constraint).

The outgoing Request messages are then sent to the selected ASs, thus advancing the path discovery process.

Now, depending on the network topology, intermediate ASs can receive several Request messages with the same PCID; for instance, AS 2 and AS 3 may both receive two Requests, from AS 1 and AS 3, and from AS 1 and AS 2 respectively. However, only the *first* received Request (i.e., the one carrying a *new* PCID) is processed and forwarded; subsequent Requests related to the same PCID are instead stored into the database, but they do not trigger any further transmission; in other words, *at most one Request is sent between two neighboring agents for a given path computation*.

All the Requests that have been successfully sent to (and acknowledged by) the neighboring ASs are stored into a local database, in order to keep track of the expected responses. If the transmission of a Request message fails for any reason (e.g. network error, or the peering agent is unable to accept the message), such Request is not stored into the database. If an AS is unable to forward a Request message (e.g., because no TE-feasible route exists from itself to the destination), then it immediately notifies the neighbors from which it has received a Request that its branch of the path computation has failed. This is accomplished by means of the Error message. The latter includes an *ID Object* and an *Error code*, i.e. an informative field which specifies the reason of the failure.

Path Collection: Assume that the order of sent Requests is the one shown in Figure 1. AS 4, the Dest AS, receives two Requests, one from AS 2 and one from AS 3, which concludes the Path Discovery phase. At the destination AS, incoming Requests are processed as they arrive. For each received message, local policies are checked to determine whether an LSP originating from the source AS is allowed, and whether the bandwidth and QoS constraints are verified (this time only taking into account any downstream contribution, of course). If not, an Error is sent back with the appropriate error code. Otherwise, a Response message is prepared and sent back to the sender, thus starting the Path Collection phase. The *ID Object* is followed by one or more *AS-Path Information Objects*, each of them describing the characteristics of the partial AS path to the destination. The *BW* parameter is a copy of the corresponding value received in the Request. The *CumQoS* field contains the cumulative QoS of each partial AS path in the Response. The destination AS initially sets it according to its local QoS parameters. Response messages originate at the destination and travel backwards to the source along the *same* AS paths traversed during the Path Discovery phase.

When an intermediate AS receives a Response from a neighbor, it stores it into the database, and updates each AS-Path Info Object: it adds its own AS number to the PATH and updates the *CumQoS*, including its own contribution. Note that, unlike the *Residual QoS Budget* in the Requests, the *CumQoS* represents the cumulative QoS for the sub-path starting from the current AS down to the destination. However, the intermediate AS does *not* necessarily reply immediately to its upstream neighbors. Instead, it adds the updated *AS-Path Info Objects* to the set P of available AS-Path results already received for the specific *PCID*, and checks whether *all* the Requests that it has sent out during the Path Discovery phase have been replied to (through either a Response or an Error). If not, it simply *waits for more messages to arrive* (recall that the IA-PCP objective is to harvest path diversity as much as possible).

If all the required Responses/Errors have been received, the agent examines the set P, and extracts from it a subset $p \subseteq P$ of up to RES_{MAX} AS paths, where RES_{MAX} is a local preconfigured value. Such a filtering is required to prevent the number of AS-Paths returned to the source AS to explode exponentially as the contents of more and more Responses are assembled on the way back to the source. Possible AS path ranking functions for constructing p are:

- *FIFO*: each AS just selects the first RES_{MAX} AS paths. Besides being simple, this makes sense as it favors quick neighbors. If a path comes back quickly, it is also likely to yield a good delay performance.
- *Shortest AS path*: paths are sorted by AS path length.
- *QoS*: paths are sorted by a QoS metric, so that the one with the best QoS to the destination comes first.
- *Hybrid*: some of the above criteria (e.g., QoS metrics and AS path length) are combined through a weighted average.

As already said, the source can specify a preferred path

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7262
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7262

ranking function in its Request message. Intermediate ASs along the path are not allowed to change this requirement in the Request and Response messages. However, they can use a different path ranking function if the required one is not available, or due to local policies taking priority over global ones. The subset p is the only one propagated backwards to the upstream neighbors. The IA-PCP agent retrieves the list of upstream neighbors from which a Request was received related to the current path computation, and sends them a new Response containing the filtered subset p of AS-Path Info Objects, if the latter is non empty, or an Error message otherwise. Note that "late" Requests arriving at an AS after a Response has already been issued can be replied to immediately using the same Response/Error message. We observe that at most one Response or Error message is sent back to a neighbor AS.

The Path Collection phase terminates when the source domain receives a Response or an Error for each sent Request. The source domain possesses a list of AS-Paths satisfying all the constraints specified in the original request.

In the example, when AS 4 receives the Request from AS 2 it replies with a Response containing exactly one AS-Path Info Object (Path: 4), and the same is sent in reply to AS 3. In turn, AS 3 sends a Response to AS 1 (Path: 3 4) and to AS 2 (Path: 3 4), and AS 2 also sends a Response to AS 1 containing two AS-Path Info Objects: (2 4) and (2 3 4). Finally, AS 1 receives the Responses from AS 2 and AS 3 and concludes the computation collecting three AS-Paths: (1 2 4), (1 3 4) and (1 2 3 4). Note that (1 2 3 4) is not in AS 1's BGP routing table.

Path Commit/Cancel: The IA-PCP protocol includes a third phase, namely the Path Cancel/Commit phase. A *Cancel* message , consisting in the sole *ID Object* field, is sent to all the ASs to which a Request was sent and it is used to notify the agents involved in a computation to discard the state related to a given path computation and to neglect further messages related to a given PCID. Each intermediate AS simply forwards it to all the downstream ASs it sent a Request to for that PCID. Depending on the architectural context in which IA-PCP is employed, a *Commit* message can also be issued by the source AS. The latter travels along the AS path selected by the source AS (instead of the Cancel message) and is meant to inform the involved agents that a PCE-assisted path computation is attempted through that AS path. If IA-PCP agents are employed solely for AS path computation, a *Commit* message is not necessary. On the other hand, in [18] IA-PCP is used as part of a larger TE framework which integrates PCE computation and RSVP sessions in a unique tool. In that case, the Commit message is sent after a successful LSP setup through RSVP, to notify the involved AS to update their TE database. The Commit message includes the PATH field along which it traverses.

C. *Mechanisms for Robustness*

We now describe the aspects of IA-PCP related to robustness. We have already seen that some basic loop prevention is enforced in the Request message processing. However, this is not enough to avoid deadlocks. As an example, let us consider a portion of the network comprising four domains connected as illustrated in Figure 2. Assume that an AS path computation from S to D (not shown) is started. Each of the four nodes receives one Request message (R1, …, R4) almost at the same time, i.e. before the fastest of them is able to process the message and forward its own requests downstream. Depending on how the BGP table, local policies, etc. are configured, the following sequence of events can take place: AS 1 forwards two requests, one (RF1-A) to a node outside the graph and another (RF1-B) to AS 4; AS 2 sends a request to AS 1 (RF2), while AS 3 sends RF3-A to a node outside the graph and RF3-B to R3. Finally, AS 4 forwards RF4 to AS 3. It is easy to see that no AS path loop can be detected by the test described in subsection B (see point 2). However, according to the protocol rules, when the forwarded requests are received by the respective peering domains, no action is taken locally since an earlier request for the same PCID has already been served and the agents are waiting for Responses. However, the four domains are stuck in a circular waiting list, because AS 1 is waiting for a response from AS 4, which in turn is waiting for AS 3, which is waiting for AS 2, which is finally waiting for AS 1.

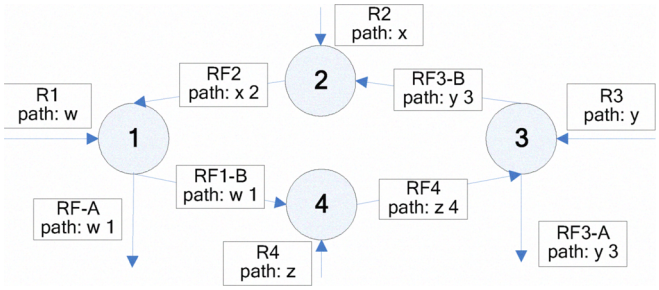

Figure 2. Deadlock scenario

To cope with such situations, timeouts are required. When a Request message is sent to a neighbor domain, a Request timer (*ReqTimer*) is armed to expire after the following time:

$$T_{\text{Req}} = nR^{(hC \cdot reqT)}$$

where hC is the hopCount value received in the Request message, nR and $reqT$ are tunable parameters of the protocol. The above formula stems from considering that the higher hopCount is, the greater the chances are that Requests will travel more, which increases the overall response time under normal operating conditions. Preliminary simulation studies (omitted due to lack of space) showed that a linear relationship between T_{Req} and hC often leads to frequent timer expirations on the most upstream ASs, hampering the protocol effectiveness. If the timer expires, meaning that no Response messages have been received after T_{Req} seconds, an Error message with the appropriate error code is sent to all the upstream ASs. This allows the Path Collection phase to progress across the network also in the presence of deadlocks or crashes of IA-PCP agents.

The ReqTimer is stopped when the first Response is received. At that time, the IA-PCP agent arms another timer (*ResReadyTimer*), whose expiration time is set similarly as above:

$$T_{\text{ResReady}} = nR^{(hC \cdot resT)}$$

where $resT$ is a tunable time constant, smaller than $reqT$. The ResReadyTimer is stopped when all the necessary Responses or Error messages have been received. When it expires, a Response is assembled with the information received so far from the downstream domain and propagated backwards. The purpose of this second timer is to improve efficiency as well as robustness. In fact, when some Responses have already been collected, it is reasonable not to overly delay the AS path com-

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7262
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7262

putation waiting for the last possible message. Furthermore, large difference in the response time from two neighboring ASs are likely to be accountable to similar differences in the returned AS path lengths, which adds to the point.

D. Discussion

We outline here some properties of IA-PCP. First of all, we compute a bound on the number of messages generated by IA-PCP. An AS i contacted for an AS path computation may generate a maximum number of Request messages equal to:

$$r_i \le \min\{H_i, S_i, R_i\}$$

where H_i is the number of distinct next-hop ASs, S_i is the number of SLA established with neighbors allowing interdomain TE tunnels, and R_i is the configured REQ_{MAX} value at AS i. As each Request triggers at most one Response/Error and one Cancel/Commit message, the maximum number of messages generated by AS i is $M_i \le 3 \cdot r_i$. The number of received Requests cannot be limited through administrative constraints. However, the overall number of received *and* generated Requests is upper bounded by H_i. Thus, at most *three* messages per path computation traverse a given interdomain link. The bandwidth used by IA-PCP is also negligible: for a Response message (i.e., the largest, due to the included AS paths), it takes 50 paths of 10 AS hops each to make up 1kB, since AS numbers are 16 bit values. Moreover, each AS also is allowed to limit the number of returned paths. The above numbers show that bandwidth consumption is not an issue with IA-PCP.

The processing overhead at each IA-PCP agent does not represent an issue either. The required database accesses (e.g., to BGP RIBs, to a policy database to examine SLAs, to a local database maintaining ingress-to-egress TE and QoS information) are related to relatively stable (and thus easily cacheable) information, and can be managed in few milliseconds (see [18] for some measurement of the times required by IA-PCP). Furthermore, the complexity involved in the AS path ranking function sorting is at most $O(H \cdot k \log k)$ for an AS selecting k paths from H neighbors, each one reporting k paths itself. Given the actual degree of connectivity of Internet ASs, such orders of magnitude do not represent a problem.

An upper bound to the time it takes for AS i to respond to a Request can be recursively computed as:

$$T_i = T_C + \min\left[\max_{j \in H}(T_j), \min_{j \in H}(T_j) + T_{\text{ResReady}i}, T_{\text{Req}i}\right],$$

where T_C is the time spent in local computations, T_j is the service time taken by each neighboring AS to which the request has been forwarded. Note that, since requests are sent in parallel to all neighbors, their respective computation times do not add up. The global response time for a path computation is not easy to evaluate analytically, and it obviously largely depends on the constants selected for the timeouts. In Section IV we report a preliminary evaluation, showing that good results can be obtained in reasonable time (in the order of few tens of seconds) for paths up to nine AS hops.

Note that IA-PCP does not require that ASs expose their internal topology, TE resources or QoS. As communications are only bi-lateral, and only take place among neighbors which have already established mutual relationships, IA-PCP poses no significant security problems, and it is sufficient that IA-PCP agents be mutually authenticated.

We observe that caching the results of AS path computations performed on behalf of other ASs may enable a domain both to reply in a quicker way to subsequent requests using cached information, and to steer the search for TE-feasible AS paths more effectively based on historical records, generating and processing fewer messages. Further study is needed to weigh the overhead of storing and managing the above information with the possible benefits, also keeping into account the risk that effectiveness be hampered by stale information.

IV. Performance Evaluation

In this section, we show how IA-PCP can compute a good, close to optimal, set of AS paths in reasonable time. We implemented IA-PCP in the ns2 simulator [29].

A. Scenario and metrics

The test scenario has been generated using the BRITE topology generator [30]. The scenario has 500 ASs, each one represented as a single node. The settings are the "AS-only" standard BRITE settings, i.e. Waxman topology, with $\alpha = 0.15$, $\beta = 0.2$, $m = 2$ and TE capacity at the interdomain links taken from a uniform distribution between 10 and 1024. We assume that intradomain TE capacity does not represent a constraint for path setup. As a QoS metric, we simulate the delay of each AS, which is split into an intradomain and an interdomain part. The former is assumed for simplicity to be constant for all intradomain paths. The interdomain part is instead link-specific. Both are taken from a uniform distribution between 0.1 and 5.0 ms. The computation time at each IA-PCP agent is taken from a uniform distribution between 120 and 200 ms, the latter fitting the times actually measured in the IA-PCP implementation within EuQoS. In that scenario, we first run BGP so as to populate the routing tables in each AS. We select source and destination ASs with different degrees of connectivity, and we perform a number of AS path computations between them.

Our purpose is to assess how good the set of AS paths returned by IA-PCP is. The problem of evaluating a *set* of paths (rather than a single one) bears some considerations. Capitalizing on knowing the simulated topology exactly, we first compute the set of AS paths to a destination in a centralized way: we perform a recursive search for *all* the paths to a destination up to a large enough maximum distance (e.g. 10 hops), subject to the same constraints (TE capacity, QoS) as in the IA-PCP request. Then, we sort these paths according to the selected path ranking function, and store the ones "close enough" to the optimum in a test set Ts. "Close enough" means within a 20% end-to-end delay or a 2 hop margin with respect to the best AS path, depending on the AS path ranking function. We assign a score equal to $|Ts|/j$ to the j^{th} AS path in Ts. Then, for each AS path computation, we run IA-PCP. The set of returned paths Tp is tested against Ts: we give the same score to each AS path in $Ts \cap Tp$, and a null score to those in $Tp \setminus Ts$, and we compute the *weight* of Tp, $w(Tp)$, as the ratio of its total score to that of Ts. Clearly, $w(Tp) = 1 \Leftrightarrow Ts \subseteq Tp$, whereas the larger $Ts \setminus Tp$, the smaller $w(Tp)$, all the more if the paths in $Ts \setminus Tp$ are the top-ranking ones.

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7262
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7262

B. Results

All the reported results are averages of nine IA-PCP computations between different sources and destinations. Figure 3 shows the computation time and the number of request messages as functions of the initial hopCount, with *nR*=2s, *reqT*=2 and *resT*=1. First of all, we observe that the path computation time grows roughly exponentially with the initial hopCount, reaching about two minutes for nine hops. According to [32], paths of more than seven AS hops are rare in the Internet, which should make the computation time of IA-PCP reasonable also on a larger scale. Furthermore, the number of request messages does not grow indefinitely, but tends to cap for hopCounts larger than seven. This suggests that both the *ReqTimer* and the *ResReady* timer can be capped as well, to prevent computations to last arbitrarily long due to misconfigured hopCount values in the source domain Request message. Hereafter, we set the initial hopCount to eight.

We then evaluate the computation time and $w(Tp)$ as a function of the timer constants *reqT* and *resT*. Each computation requests 100 Mbps of bandwidth. RES_{MAX}, is set to 30, and paths are ranked by their length. Figure 4 shows $w(Tp)$ and the computation times as a function of *resT* , for two values of the *reqT* constant. The figure shows that the performance is insensitive to *reqT*, which suggests that deadlocks happen infrequently, and they do not delay the computation. On the other hand, $w(Tp)$ initially increases with *resT*. The computation time, instead, heavily depends on *resT*, growing exponentially. This suggests that an optimal *resT* value can be computed, trading off between effectiveness and response time. We remark that inter-AS TE tunnels are not expected to be negotiated frequently, as they can be thought of as semi-permanent connections established as part of a provisioning process. For that reason, computation times in the order of tens (or even few hundreds) of seconds are indeed tolerable when coupled with a well-founded expectation of improved performance.

Finally, we evaluate $w(Tp)$ as a function of RES_{MAX}, setting *nR*=2s, *reqT*=2 and *resT*=1. Figure 5 and Figure 6 report the results for AS paths ranked by length and by QoS respectively. In both cases, $w(Tp)$ is always above 0.7, and it increases, although not much, with RES_{MAX}. We observe that $w(Tp)$ is almost always increasing with the required bandwidth (except for one case in Figure 5). In fact, most paths have a minimum bandwidth between 200 and 270 Mbps. For higher requests, fewer paths make it into *Ts* (from an average of 50 for 100Mbps to 19 for 300Mbps in the case of Figure 5, and from 5.1 to 1.8 for Figure 6), so that it is more likely for IA-PCP to pick all (or most) of them.

V. Implementation of the IA-PCP within EuQoS

The IA-PCP has been implemented in a fully functional prototype developed within the framework of the IST-EuQoS project [10], which also includes the IETF PCE architecture. In the above architecture, the IA-PCP is run by the *Traffic Engineering and Resource Optimization (TERO)* module. The latter manages service level agreements with the neighboring domains, configures the BGP decision process, and it also provisions resources (buffer, bandwidth, policing) at the inter-domain links. The setup of an inter-AS MPLS-TE tunnel is requested by the system operator of the head-end AS, which issues a request between two remote peers via the TERO web interface. The head-end TERO module runs the IA-PCP, and comes up with a list of AS-paths. It then acts a PCC, and requests a path computation to the PCE, specifying the selected AS-path as an *Implicit Route Object*. Once the PCE response comes back, the ERO is passed to the to the ingress LSR, that starts the RSVP-TE session to actually setup the path. Within EuQoS, inter-AS tunnels are provisioned to transport QoS-guaranteed connections between non-neighboring ASs. This spares the transit ASs traversed by the tunnel from handling the possibly large amount of signaling messages required for controlling the admission of, and reserving resource for each single flow. The above architecture has been installed on the EuQoS pan-European testbed, consisting of twelve local testbeds connected to the respective National Research and Education Networks (NRENs) and interconnected through the GEANT European backbone networks. Measurements taken on the testbed [16], [33] showed that the time taken to setup an interdomain tunnel spanning four AS (therein including IA-PCP, PCE computations and RSVP-TE setup) is in the order of one minute.

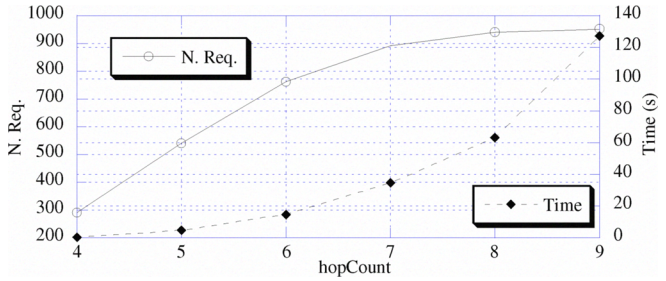

Figure 3. No. of Requests and computation time as a function of the initial hopCount

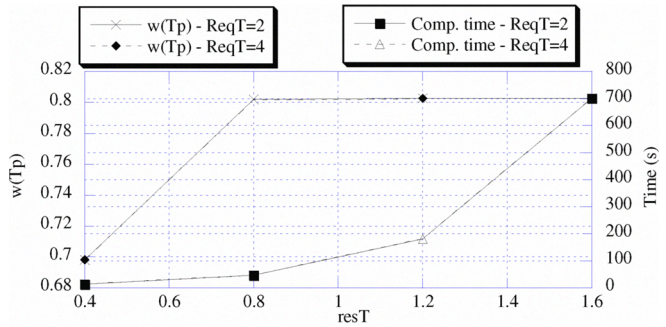

Figure 4. $w(Tp)$ and computation time as a function of *resT*

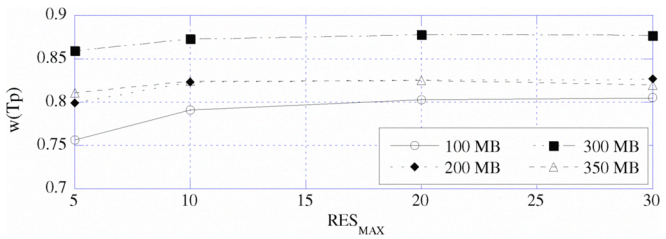

Figure 5. $w(Tp)$ as a function of RES_{MAX} – Ranking by AS Path length

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7262
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7262

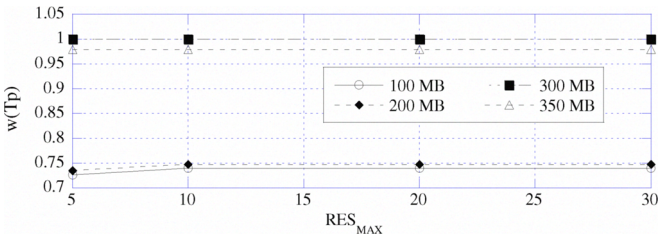

Figure 6. $w(Tp)$ as a function of RES_{MAX} – Ranking by QoS

VI. Conclusions and Future Work

Moving from the concern that selecting a good AS path is important for interdomain Traffic Engineering, and that BGP routes are often too few and not necessarily suitable to this purpose, this paper has described the Inter-AS Path Computation Protocol (IA-PCP). The latter does not require centralized computation, but relies instead on the exchange of few messages between neighboring ASs, which cooperate to report to the requesting AS a number of AS paths to a given destination. Path computation can be constrained by bandwidth and QoS metrics. Each AS participating in a path computation selects a subset of the paths it learns, in order to avoid combinatory explosion. The results show that, if the selection is made coherently by all ASs, practically good results can be achieved in reasonable time. There are several directions to extend this work. First, an extensive evaluation of IA-PCP is needed, both by simulation and using the already deployed testbed, in order to calibrate the timers. Second, the present framework lends itself to devising heuristics for path ranking and selection, which may blend or compose the already presented criteria. Finally, as IA-PCP provides a source with *several* paths, we are considering devising heuristics to capture path diversity, which could be also used in conjunction with multipath routing solutions.

VII. Acknowledgements

As this work was started within the framework of the IST-EuQoS project, we would like to thank all the EuQoS consortium partners for their contribution to discussion in its early stage. Specific thanks go to Olivier Dugeon of France Telecom R&D. Finally, we thank Daniele Ribolini for helping us with the simulations.

References

[1] N. Wang, K.H. Ho, G. Pavlou, M. Howarth, "An Overview of Routing Optimization for Internet Traffic Engineering", IEEE Communications Surveys and Tutiorials, 10/1, 2008, pp. 36-56

[2] RFC 4202, Routing Extension in Support of Generalized Multi-Protocol Label Switching (GMPLS), Oct. 2005.

[3] RFC 4655, A Path Computation Element (PCE)-Based Architecture, Aug. 2006.

[4] RFC 4657, Path Computation Element (PCE) Communication Protocol Generic Requirements, Sep. 2006.

[5] RFC 4674, Requirements for Path Computation Element (PCE) Discovery, Oct. 2006.

[6] JP. Vasseur, *et al.* "A Backward Recursive PCE-based Computation procedure to compute shortest inter-domain Traffic-Engineering Label Switched Paths", draft-ietf-pce-brpc-09, Apr. 2008.

[7] RFC 5152. "A Per-Domain Path Computation Method for Establishing Inter-Domain Traffic Engineering (TE) Label Switched Paths (LSPs)", Feb. 2008

[8] R. Bradford, *et al.* "Preserving Topology Confidentiality in Inter-Domain Path Computation using a key based mechanism", draft-ietf-pce-path-key-06, March 2009.

[9] L. Fang, N. Bita, J.-L. Le Roux, J, Miles, "Interprovider IP-MPLS Services: Requirements, Implementations, and Challenges", IEEE Communications Magazine, June 2005, pp. 119-128

[10] The IST-EuQoS project, http://www.euqos.eu

[11] O. Dugeon *et al.*, "End to End Quality of Service over Heterogeneous Networks", in Proc. NetCon'05, Nov. 2005.

[12] K. Knightson, *et al.* "NGN Architecture: Generic Principles, Functional Architecture, and Implementation", IEEE Com. Mag., Oct. 2005, pp.49-56.

[13] W. Xu, J. Rexford, "MIRO: Multi-path Interdomain ROuting", SIGCOMM'06, Pisa, Italy, Sep. 11-15 2006.

[14] RFC 4726, A Framework for Inter-Domain Multiprotocol Label Switching Traffic Engineering, Nov. 2006.

[15] RFC 4216, MPLS Inter-Autonomous System (AS) Traffic Engineering, Nov. 2005.

[16] O. Dugeon *et al.* "Prototype P#4 tests report", Deliverable 5.2.3, IST/FP6 EuQoS consortium, December 2007, http://www.euqos.eu

[17] A. Farrel *et al.*, communications on the mailing list of the IETF Path Computation Element Charter, 21 March – 7 April 2008

[18] O Dugeon, E. Mingozzi, G. Stea, L. Bisti, "Provisioning QoS in Interdomain Traffic Engineering", Springer Annals of Telecommunications, 2008

[19] A. Sprinston, M. Yannuzzi, A. Orda, X. Masip-Bruin "Reliable Routing with QoS Guarantees for Multi-Domain IP/MPLS Networks", in Proc. IEEE INFOCOM 2007, Anchorage, Alaska, USA, May 2007

[20] S. Secci, J.-L. Rougier, A. Pattavina, "On the Selection of Optimal Diverse AS-Paths for Inter-Domain IP/(G)MPLS Tunnel Provisioning", Proc. of IT-NEWS 2008, pp. 235-241, 13-15 Feb. 2008, Venice, Italy.

[21] R. Douville, J.-L. Le Roux, J.-L. Rougier, S. Secci, "A Service Plane over the PCE Architecture for Automatic Multidomain Connection-Oriented Services", IEEE Communications Magazine, June 2008, 94-102

[22] D. Walton *et al.* "Advertisement of Multiple Paths in BGP" draft-walton-bgp-add-paths-06.txt, exp. date Apr. 2009

[23] F. Aslam, *et al.* "Inter-Domain Path Computation using Improved Crankback Signaling in Label Switched Networks," Proc. IEEE ICC '07, Glasgow, U.K., June 2007, pp. 2023-2029

[24] B. Huffaker, M. Fomenkov, D. J. Plummer, D. Moore, K. Claffy, "Distance metrics in the Internet", *IEEE Intern'l Telecommunications Symposium*, 2002

[25] C. Pelsser, O. Bonaventure, "Path Selection Techniques to Establish Constrained Interdomain MPLS LSPs", in Proc. of IFIP Networking Conference, 15-19 May 2006, Coimbra, Portugal, LNCS 3976

[26] F. Dabek, R. Cox, F. Kaashoek, and R. Morris. "Vivaldi: A decentralized network coordinate system", in Proc. of ACM SIGCOMM'04 Portland, Oregon, August 2004.

[27] X. Yang, d. Clark, A. W. Berger, "NIRA: A New Inter-Domain Routing Architecture", IEEE/ACM Transactions on Networking 15/4, August 2007, pp. 775-788

[28] M. O. Howarth et al., "End-to-end quality of service provisioning through inter-provider traffic engineering", Elsevier Computer Communications 29 (2006), pp. 683-702

[29] The Network Simulator, http://www.isi.edu/nsnam/ns/

[30] BRITE, http://www.cs.bu.edu/BRITE/

[31] ITU-T Rec. Y.1540, "Internet protocol data communication service - IP packet transfer and availability performance parameters," Nov. 2007.

[32] B. Quoitin, S. Uhlig, C. Pelsser, L. Swinnen and O. Bonaventure, "Interdomain Traffic Engineering with BGP", IEEE Com. Mag., May 2003, pp. 122-128

[33] E. Mingozzi, G. Stea, *et al*, "EuQoS: End-to-End Quality of Service over Heterogeneous Networks", *Elsevier Computer Communications*, to appear (2009)

Digital Object Identifier: 10.4108/ICST.BROADNETS2009.7262
http://dx.doi.org/10.4108/ICST.BROADNETS2009.7262