Inter-Domain Path Computation using Improved Crankback Signaling in Label Switched Networks

Faisal Aslam
Freiburg University
Germany
aslam@informatik.uni-freiburg.de

Zartash Afzal Uzmi
Computer Science Department
LUMS, Pakistan
zartash@lums.edu.pk

Adrian Farrel
Old Dog Consulting
United Kingdom
adrian@olddog.co.uk

Michal Pióro
Institute of Telecommunications,
Warsaw University of Technology
mpp@tele.pw.edu.pl

***Abstract*—For label switched networks, such as MPLS and GMPLS, most existing traffic engineering (TE) solutions work in a single routing domain. These solutions do not work when a route from the ingress node to the egress node leaves the routing area or the autonomous system (AS) of the ingress node. In such cases, the path computation problem becomes complicated because of the unavailability of the complete routing information throughout the network. We present CWS (computation while switching), a new inter-domain path computation scheme which tries to compute a near-optimal path without assuming the availability of complete topology information. We provide a detailed comparison of the CWS scheme with another per-domain path computation scheme given in [1].**

Unlike the standard per-domain path computation scheme [1], the CWS scheme continues the quest for a better path instead of terminating the search at the first available path, resulting a significant improvement in terms of path optimality. In particular, CWS guarantees that, for a given network state, a computed inter-domain path will traverse a minimum number of domains. This improvement in path computation directly impacts the amount of traffic that can be allowed on the network. For example, for the COST266 topology with 28 domains and 37 bidirectional inter-domain links, CWS places 960 of the requested 2000 paths as compared to 683 paths placed by existing schemes. Finally, the path setup latency of the CWS scheme remains comparable to that of existing schemes, by allowing the data flow as soon as the first feasible path is found.

I. Introduction

The emergence of mission-critical and other multimedia applications such as VoIP, videoconferencing, e-commerce, and VPNs has translated into stringent real-time QoS requirements for carrier networks. A key factor in meeting the QoS requirements of such applications is the ability to route traffic along explicit paths computed through constraint based routing. The destination-based, shortest-path routing paradigm employed in IP routing does not support routing network traffic along explicit paths. However, the emergence of label switching paradigms such as MPLS and GMPLS has overcome this limitation by presenting the ability to establish a label switched path (LSP) between two points on an IP network. This ability to do traffic engineering (TE) using MPLS maintains the flexibility and simplicity of an IP network while exploiting the advantages of an ATM-like connection-oriented network.

Ingress routers of an MPLS network classify packets into forwarding equivalence classes (FECs) and encapsulate them with labels before forwarding them along pre-computed paths. The path a packet takes as a result of a series of label switching operations in an MPLS network is called a label switched path (LSP). Label switched paths (LSPs) may be computed by using a constrained shortest path first (CSPF) algorithm which essentially finds a shortest path between two network nodes subject to constraints such as maximum delay, minimum available bandwidth, and resource class affinity. Thus, the constraints dictate how the traffic should be engineered through the network and, for this reason, the paths computed under the constraints are called traffic engineered paths (TE paths). For the computation of a TE path that an LSP would traverse, a CSPF algorithm uses the TE information, such as the remaining or reserved bandwidth along a link, advertised throughout the network. Resources along a computed TE path are reserved simultaneously with label distribution, using protocols such as CR-LDP and RSVP-TE [2].

Existing solutions for traffic engineering in MPLS and GMPLS networks are mostly limited to work within a single routing domain[1] and do not work when traffic leaves the boundaries of the routing area or autonomous system (AS) of an LSP ingress node [3], [4]. This is primarily because, in such a case, the ingress node has limited visibility of the topology and TE resource information (e.g., bandwidth, delay), outside its own routing domain and, therefore, can not compute an end-to-end optimal path that spans multiple domains. An optimal inter-area path refers to a constrained-shortest path that would be computed as if there is a single routing domain and complete topological and resource information is available during computation [5]. In addition, one may impose an extra constraint of minimizing the number of domains traversed by the computed route [5].

We present a new inter-domain path computation scheme called Computation While Switching (CWS), which tries to compute an optimal or near-optimal path while assuming availability of limited topological information and fulfilling a subset of inter-domain path computation requirements. Such requirements include: (i) No advertisement of internal topology or resource information outside domain boundaries, which

[1]The term "routing domain", as used in this paper, refers to a network under a single administration with common policies. A routing domain, therefore, may be an autonomous system or a routing area within the autonomous system.

is critical for security, confidentiality and preservation of the scalability of routing protocol used within a routing area (the Interior Gateway Protocol, IGP) and the routing protocol that exchanges information between the networks (e.g., BGP). (ii) No unreasonable increase in IGP load and preservation of RSVP scalability [3], [4].

The CWS scheme fulfils these requirements and shows significant improvement as compared to the scheme in [1], in terms of resulting path optimality. In particular, for a given network state, an inter-domain path computed using CWS will traverse a minimum number of domains. Furthermore, the worst-case path setup latency of CWS remains the same as that of the scheme in [1], and may further be reduced using the optional *Stop-And-Wait-Error* procedure included in CWS.

The rest of the paper is organized as follows. Section II and III provide the relevant background material. Existing inter-domain path computation schemes are given in section IV while section VI details our evaluation methodology. We provide the details of CWS scheme including the optional *Stop-And-Wait-Error* procedure in section V-B. Simulation setup and results are discussed in section VII and we conclude in section VIII.

II. Path Setup and Information Scenarios

In a network that employs traffic engineering (TE), destination based hop-by-hop forwarding may be replaced by sending traffic along explicit paths from ingress to the egress node. An initial explicit path is usually computed by the ingress node which creates a corresponding explicit route object (ERO) that simply enlists the network elements along the computed path. Due to confidentiality constraints across routing domains, an ingress node does not usually have the capability to compute a hop-by-hop path to the egress. Instead, a path that includes some, but not all, intermediate nodes to the egress is computed and included in the ERO. Along this route, also referred to as a *loose roue*, the path setup request is sent as an RSVP *Path* message. This request along the loose route also includes the ERO. An intermediate node may have to expand the ERO, received in *Path* message, thereby refining the loose route by specifying more intermediate nodes based on the local domain information.

The optimality of a computed constrained path and how loosely that path is specified depends upon the amount of network information available at the ingress node. This leads to the definition of three information scenarios as given in [5]: Multi-Domain Visibility, Partial Visibility, and Local Domain Visibility. These information scenarios impact the optimality of the computed path and are summarized below:

A. Multi-Domain Visibility

In this scenario, the ingress node has sufficient information about the topology and TE resources of all relevant domains. Hence, the ingress node can compute a complete end-to-end inter-domain optimal path. Thus, the ingress node can fully specify and signal an explicit, end-to-end optimal path, in the ERO, from itself to the egress node. However, achieving this information scenario may not be practical given the requirement that a domain should never advertise TE resources and topological information outside its boundaries [3], [4].

B. Partial Visibility

In this scenario, the ingress node has full information about its own domain, and has information about the connectivity between the domains as well as the TE resources availability across the other domains, but it does not have full visibility of the topologies *inside* other domains. Consequently, the ingress node is not able to provide, in the ERO, a fully specified strict explicit path from ingress node to the egress node. However, the ingress node can still supply some useful information about intermediate domains, and signal this within the ERO. For example, the ingress node might supply an explicit path that comprises:

1) explicit hops from the ingress node to the local domain boundary
2) intermediate domains represented as *abstract* nodes or their entry points specified as loose hops, where an abstract node refers to a group of nodes whose internal topology is opaque to the ingress node of the LSP, e.g., a domain other than the domain of the ingress node
3) a loose hop identifying the egress node.

Note that it is possible that the ingress node does not have information about the TE resources of other domains in which case the ingress node can try multiple paths one-by-one in order to successfully place the LSP along a path that satisfies the TE constraints.

C. Local Domain Visibility

The ingress node has full visibility of its own domain and connectivity information only as far as determining one or more domain border routers (DBRs)—routers that connect two routing domains[2]—that may be suitable for carrying the LSP to its egress node. In this case the ingress node builds an explicit path that comprises just:

1) explicit hops from the ingress node to the local domain boundary
2) a loose hop identifying the egress node.

III. Crankback Signaling

Crankback signaling facilitates rerouting of an LSP setup request, around blocked or failed network elements, in case a path setup attempt is unsuccessful [6]. Thus, crankback signaling is a way using which a router indicates if it could not reserve resources along a TE path to the egress router. Crankback signaling, which enhances the existing RSVP-TE signaling to establish LSP tunnels [2], [7] has additional applications including inter-domain constrained path computation and restoration routing in the case of multiple network element failures.

In case of an unsuccessful path setup attempt (i.e., unsuccessful RSVP-TE *Path* message [2]), crankback information

[2]Examples of DBR include IGP area border routers (ABRs) or Autonomous System border routers (ASBRs).

from a failed or blocked network element is propagated upstream using *PathErr* or *Notify* messages. This information includes the identity of the blocked or failed network element and the reason for path establishment failure, for example, congestion, network element failure, or resource violation.

Upon receiving a *PathErr* message, an upstream routing point[3], say RCOMP, saves the crankback information in its local crankback history table and discards the *PathErr* message. Subsequently, it attempts to reroute, avoiding the blocked elements specified in its crankback history-table. The routing point updates the local crankback history-table, with the latest crankback information after each failed path setup attempt, so that any successive rerouting attempts can avoid all known blocked network elements. A routing point may opt to discard the information stored in the local crankback history-table after an LSP is successfully established.

An RSVP *PathErr* message is generated by RCOMP if it is unable to successfully establish a path using any of its domain exit points. This *PathErr* message contains the summarized information from the history-table of RCOMP and is sent to the upstream nodes, such that an upstream routing point (further upstream from RCOMP) can benefit from the experiences of RCOMP. Furthermore, the upstream routing point should also avoid rerouting attempts through RCOMP because now it is established that no path exists through it.

The ingress node uses the LSP_ATTRIBUTE object of the *Path* message to nominate the nodes that may perform the role of a routing point during the computation of a complete path. For this purpose, four rerouting flags are suggested [8]: (1) *No Rerouting*: Only the ingress node may attempt to reroute; intermediate nodes may or may not supply crankback information. (2) *End-to-End Rerouting*: Only the ingress node may attempt to reroute, while intermediate nodes must supply crankback information. (3) *Boundary Rerouting*: Only intermediate DBRs or the ingress may attempt rerouting after receiving crankback information, while other nodes should supply crankback information. (4) *Segment-based Rerouting*: Any node may attempt rerouting after receiving crankback information. If a given node chooses not to perform rerouting then it should still supply crankback information upstream.

IV. Existing Path Computation Schemes

Inter-domain path computation schemes can be divided into two categories: PCE based computation and per-domain path computation. Depending on the service provider requirements and functionality available on nodes, one may adopt either one of these techniques. Our proposed CWS scheme falls under the per-domain category and, therefore, we restrict the comparison of CWS to the existing per-domain path computation scheme. We also provide a brief account of both types of path computation schemes:

[3]The term routing point refers to a label switched router (LSR) that can perform path computation. There may be several routing points along the complete path from ingress to egress.

A. PCE based Path Computation

A Path Computation Element (PCE) is an entity—a node or a process—that is responsible for computing an inter-domain TE-LSP upon receiving a request from a Path Computation Client (PCC) which could also be another process or a node. There could be zero, one or more PCEs per domain, and a PCE may or may not reside on the same node as its corresponding PCC. A path may be computed either by a single PCE node or a set of distributed PCE nodes that collaborate during path computation [9].

A PCE may compute the end-to-end path itself if enough topology and TE information is available. Alternately, it may opt to compute a part of complete path and request another PCE to compute another part [10]. Route concatenation procedure is then applied upon receiving a path computation reply from another PCE [9]. This paper restricts its consideration to per-domain path computation where the nodes along the loose route are responsible for path computation.

B. Per-domain Path Computation

Per-domain path computation is carried out by routing points along the path from ingress to the egress. As discussed in section II-A, visibility of inter-domain topology is practically limited at the ingress nodes which will only be able to determine a loose route to the egress node, unless the computed path does not need to leave the routing domain. Therefore, the portion of path through each domain must be computed by a node within that domain. This path computation mechanism, known as per-domain path computation [1], may require a series of per-domain path computations within each domain through which a loose route traverses.

A routing point RCOMP refines a loose route, received in an ERO, in the current domain such that all nodes within the current domain are identified. In doing so, RCOMP may notice that the ERO contains the domain exit point from the current domain. In this situation, local domain visibility allows RCOMP to determine a constrained path to that exit point, and thus the ERO can be refined and processed within the current domain using the normal procedure given in [2]. If, however, RCOMP discovers that the ERO does not include the exit point from the current domain, it uses crankback signaling and a configuration or an auto-discovery mechanism. In auto-discovery, a domain exit point, that can lead to the egress node, is found using IGP and BGP. This mechanism is used when the egress node is not reachable using an intra-domain path and the exit-point from the current domain cannot be deduced from the ERO, a situation that happens when the routing points have only local domain visibility.

When the ERO does not include the exit point from the current domain, the next-hop specified in the ERO will either be a loose hop DBR or an abstract node. In this case, a routing point RCOMP needs to contact a traffic engineering database (TED) which is simply a database that collects, maintains, and updates the TE information (e.g., link attributes) obtained from routing advertisements. The routing point RCOMP consults a

locally maintained TED and takes an appropriate action, as given below:

Case 1: Next-hop is NOT present in TED. In this case, the routing point RCOMP verifies IP reachability for the next-hop (a loose hop DBR or an abstract node) using IGP or BGP, and a *PathErr* message is generated if the next-hop is not reachable. Since the next-hop is not present in the TED, the next-hop must also not belong to the current domain, otherwise RCOMP will generate a *PathErr* message. Once RCOMP establishes that the next-hop is outside the current domain and is reachable, the auto-discovery mechanism is used to find the domain exit point (a DBR along the path to the egress node) using IGP and BGP information. The auto-discovery mechanism may provide multiple domain exit point options from which the next-hop can be selected. If a path can not be successfully found using a given domain exit point, crankback signaling extensions are used for retrying with alternate domain exit points.

Case 2: Next-hop is present in TED or already found using Case-1. In this case, RCOMP checks local policy as well as the policy signaled by ingress node about the LSP setup option, that is to establish the LSP as either stitching, nesting or contiguous LSP [5], [11], [12]. Subsequently, an intra-domain part of the inter-domain LSP is established based on those policies.

In both cases above, RCOMP sends a *PathErr* message to an upstream routing point, by using crankback signaling, in case RCOMP is unable to compute a path through any of the exit-points of its domain. Otherwise, it sends *Path* message to a downstream routing point or the egress node.

V. The CWS Path Computation Scheme

We observed that the traditional per-domain path computation scheme chooses the *first* available inter-domain path that fulfils the TE constraints. Thus, the resulting path could potentially be the worst possible available path. This limitation of selecting the first path in standard per-domain path computation scheme can be overcome by using Computation While Switching (CWS) scheme. Unlike the standard per-domain path computation scheme, the CWS path computation scheme continues the quest for a better path instead of terminating the search at the first available path, by making use of a few additional crankback signaling attributes. The CWS scheme uses simple extensions to crankback signaling attributes while maintaining RSVP-TE scalability. Furthermore, it provides a mechanism to select from a set of candidate paths each of which traverses the minimum number of domains. Thus, the CWS scheme can guarantee that the resulting path traverses the minimum number of domains. Finally, the CWS scheme exhibits a path setup latency similar to that of standard per-domain path computation scheme given in [1].

A. New Crankback Signaling Attributes

The key differentiating feature of CWS is the inclusion of *KT-FLAG* (keep trying) and *SKT-FLAG* (success with keep trying) attributes, which opens up the possibility of finding a "better" path even after an inter-domain path has been successfully found. The *KT-FLAG* is part of the *Path* message; its value is set by the ingress node and remains constant when the *Path* message crosses the domain. The downstream routing points, which normally do not generate a *PathErr* message if a path is successfully found, will still produce a *PathErr* message when they see the *KT-FLAG* set in the *Path* message. The *SKT-FLAG* is part of the *PathErr* message which helps the upstream routing point determine whether the downstream routing point was successful or not in finding a path. A complete list of additional crankback attributes introduced by CWS is given below:

NDC	This attribute indicates the total number of domains crossed by a given *Path* message, before encountering a path setup failure (i.e., before a *PathErr* generated).
LNDC	This attribute puts a limit on *NDC*. That is, it indicates the maximum number of domains allowed to be traversed by a *Path* message. A *PathErr* is generated if NDC becomes greater than LNDC.
KT-FLAG	Keep Trying Flag dictates that a *PathErr* message should be generated even if a path is found successfully.
SKT-FLAG	Success with *KT-FLAG* attribute is used in the *PathErr* message to inform upstream nodes that a *PathErr* is generated even though path(s) was found successfully.
DI	Domain Identifiers (DI) is a sequential list, where an entry in the list at location l is a domain identifier visited by a *Path* message after traversing $l-1$ domains.
IDPA	Intra-domain path attribute is also a list and corresponds to a *DI*. An entry at location l of an *IDPA* specifies the quality of an intra-domain path selected within a domain at location l of corresponding *DI*.
DI-set	This attribute stores the set of *DI*s and is propagated with *PathErr* message.
IDPA-set	This attribute stores the set of *IDPA*s and is propagated with *PathErr* message.

As indicated, the presence of *KT-FLAG* and *SKT-FLAG* allows the CWS scheme to search for an improved path even after an inter-domain path is successfully found. The attributes *NDC, LNDC, DI, IDPA* and *KT-FLAG* are part of the *Path* message. The values of *KT-FLAG* and *LNDC* are set by the ingress node and remain constant, whereas *NDC, DI* and *IDPA* are updated once at each domain.

B. Improved Per-domain Path Computation

The CWS inter-domain path computation scheme makes use of the auto-discovery mechanism and LSP setup options (i.e., stitching, nesting and contiguous) in the same way as outlined in [1]. However, the CWS tries to find a better path by using new crankback signaling attributes.

To setup an LSP, the ingress node with local domain visibility creates an ERO, which enlists explicit hops from the ingress node to the local domain boundary and the egress node as a loose hop. The ingress then generates a *Path* message with *NDC* set to unity, *LNDC* set to infinity, *KT-FLAG* set to true and *DI* containing the originating domain identifier. Additionally, the information about the quality of current intra-domain route is added in *IDPA*. Upon receiving this *Path* message, an intermediate routing point checks if the *DI* list already has its domain-identifier listed at the end, otherwise it adds its domain identifier in *DI*, updates *IDPA* by adding current intra-domain path quality information and increments the *NDC*. A *PathErr* is generated by the routing point if *NDC* becomes greater than *LNDC*. If the routing point successfully finds a path to the egress node then it checks *KT-FLAG* and, if the *KT-FLAG* is set to true, sends a *PathErr* message upstream with *SKT-FLAG* set to true and *DI-set* containing the copy of the *DI* received in the *Path* message.

When a routing point (or ingress node) receives a *PathErr* message with *SKT-FLAG* set to true, it sets *LNDC* equal to *NDC*. By setting *LNDC* equal to *NDC* of the currently found successful path, we are ensuring that any different path to the egress found subsequently will be no worse than the current path in terms of number of domains traversed. At this point, a routing point should also add all *DI*s, contained in *DI-set* of *PathErr* message, in local *DI-set* and similarly, *IDPA*s contained in *IDPA-set* in local *IDPA-set*.

When a routing point has exhausted trying all (or a pre-configured number of) domain exit points while using crankback signaling, it checks if local *DI-set* is empty or not. If the local *DI-set* is empty then *PathErr* is generated with *SKT-FLAG* set to false indicating that no path to egress is available through this routing point; otherwise, *PathErr* is generated with *SKT-FLAG* set to true. In the latter case, the local *DI-set* will not be empty, and the *PathErr* message should also include *DI-set* and *IDPA-set*.

At a result of above procedure, the ingress will be able to find a set of paths, each of which may reach the egress by traversing the same minimum number of domains. After the ingress selects a path, by employing a procedure described in the next section, the ingress sends a final *Path* message with *KT-FLAG* set to false and with ERO containing abstract nodes, corresponding to *DI* of the selected path. Each intermediate routing point may perform ERO expansion based on locally cached paths. Thus, the CWS scheme results in a path that is guaranteed to traverse minimum number of domains.

C. Path Selection Procedure

The CWS scheme provides the ingress a set of candidate paths in *DI-set*, each of which is guaranteed to traverse a minimum number of domains from the ingress to the egress. The ingress node must employ a procedure to select one from this set, and for this purpose, it uses the *IDPA-set*. For each candidate path *DI*, the ingress computes the sum of all entries in the corresponding *IDPA* and chooses the one for which this sum is minimized. In case, there is a tie, a path is selected whose *IDPA* entries are minimized by only considering the ingress and egress domains. The heuristic behind this is that local domain visibility ensures better routes within the domains of ingress and egress nodes. If there still is a tie, a path is randomly selected by the ingress node.

D. Intra-domain Path Optimality Attributes

The information contained in *IDPA* entries may significantly affect the outcome of the path selection procedure and optimality of resulting path. However, using the *IDPA* data-structure, one must not propagate actual path lengths as this will violate the security and confidentially requirements of inter-area path computation [3], [4]. We propose that each *IDPA* entry should contain a fictitious number, indicating the quality of the intra-domain path as compared to the other intra-domain paths of the same domain. Hence, for a given LSP, a lower *IDPA* entry for an intra-domain path i indicates that this path is better than a path k of the same domain whose corresponding *IDPA* entry is higher. Note, however, that a value associated with *IDPA* entry gives no significant information when compared with an *IDPA* entry associated with any other domain. This approach leads to near-optimal paths, while providing sufficient abstraction in order to preserve the confidentiality and security requirements mentioned in section I.

To assign *IDPA* value to an intra-domain path, event-driven signaling between different routing points of a single domain is desired. It implies that a routing point after computing a route informs other routing points in the same domain about the quality of the route computed and corresponding *IDPA* value assigned. All routing points cache the information received, such that it can be used for assigning suitable *IDPA* values to subsequent intra-domain paths within the same domain.

An *IDPA* value is assigned based on the information received from other routing points (of the same domain) and total number of intra-domain paths possible through that domain. Note that such a value is assigned without knowledge of any future intra-domain paths for the same LSP. The total number of intra-domain paths possible through any domain, say *maxp*, is equal to $n \times (n-1)$, where n is the total number of DBR of the given domain. We define P_{maxp} as the maximum value that is required to represent *maxp* paths, such that $P_{maxp} = 2P_{maxp-1} - 1$. Then the *IDPA* value is assigned as follows:

If i is the first intra-domain path found successfully through a domain to place a given inter-domain LSP, then *IDPA* entry is set to $\frac{1+P_{maxp}}{2}$. If i is next better path from m and k is next better path from i, then the *IDPA* entry for i is taken as the mean of the *IDPA* values for k and m. If i is the best path found so far, and k is the next best path, then the *IDPA* value for i is half the *IDPA* value assigned to k. The *IDPA* values for other cases can be computed similarly. The ingress node should normalize *IDPA* corresponding to the same domain, before beginning path selection procedure. This cost assignment procedure ensures that domain confidentiality is maintained while allowing the possibility of offering a better path to an upstream routing point outside the domain.

E. Path Setup Latency Considerations

As mentioned in the previous section, there could be at most $n \times (n-1)$ intra-domain routes available within a domain α, where n is the total number of DBRs in that domain. Hence, in the worst case, the per-domain path selection procedure presented in section IV-B can produce $O(n^2)$ LSP setup failures (*PathErr* messages) within that domain. It must, however, be pointed out that this latency is catered for by allowing the data flow as soon as the first feasible path is found. A *PathErr* message is required to be propagated upstream to the ingress or nearest routing point contributing significantly towards the delay in LSP setup, besides adding extra load on the network resources. In the CWS scheme, the number of *PathErr* messages generated within each domain, in the worst case, remains the same as in case of existing per-domain scheme of [1]. However, by using our optional *Stop-And-Wait-Error* procedure described below, the number of *PathErr* messages (as well as *Path* messages) generated within a domain α can be reduced to $O(\gamma^2)$, where γ is the number of domains directly connected to α. The optional *Stop-And-Wait-Error* procedure works as follows:

Instead of computing an intra-domain route from itself to a next exit point, a routing point computes in advance all the routes from itself to all the exit points belong to a neighboring domain. Subsequently, a *Path* message is generated along one of the computed paths but with additional information about the set of exit points, say δ, through which feasible intra-domain paths exist. Note that all the exit points listed in set δ lead to the same neighboring domain.

In case a downstream routing point fails to find a route, instead of generating *PathErr* message immediately, it holds the *PathErr* message and informs other DBR mentioned in δ one-by-one to compute path, using intra-domain routing point to routing point signaling. A single *PathErr* message is generated on behalf of all DBRs mentioned in δ containing aggregated information about any routes found.

F. CWS Stopping Criteria

The CWS scheme always looks for better paths; in case it has already found a path of length L, it only looks for path shorter than L. That is, the CWS stops the search along a potential path when the length of that path exceeds L, thus speeding up the search process. Furthermore, when the ingress node has already used all of its DBRs (exit points) for searching a best path for a given request then the search stops. Using the crankback attributes, a network administrator who has an idea of the length of a valid path can also control when a search should stop.

VI. Evaluation Methodology

In this paper, we compare the standard per-domain path computation scheme of [1] with CWS by using simulations on real-world POP networks. Since the main focus is to determine optimality of inter-domain paths, intra-domain topology within each POP is less significant. Furthermore, since PCE-based schemes are architecturally different from per-domain path computation schemes, a direct comparison is not attempted. We consider a multi-domain network consisting of β domains, where each domain has on average n DBRs, m nodes and l bidirectional links. We assume that the domains are non-overlapping and there are no sub-domains. We further assume that, while computing a path, the procedure of detecting and avoiding inter-domain loops is employed [5].

The LSP requests arrive one-by-one in an online fashion at an ingress node such that there is no a priori information about future requests. The k-th LSP request is characterized by the ingress node s_k, the egress node d_k and associated bandwidth demand b_k. The ingress and egress nodes for each LSP request may or may not belong to the same domain. When they belong to different domains, the resulting LSP is an inter-domain LSP. Furthermore, the resulting LSP between two nodes belonging to the same domain may still be an inter-domain LSP if the LSP has to cross the domain boundary due to insufficient TE resources to setup an intra-domain path.

In order to serve an LSP request, a bandwidth guaranteed LSP must be setup using RSVP-TE and its crankback extensions. Our main goal is to find near-optimal inter-domain bandwidth guaranteed paths using a per-domain path computation scheme, for the set of LSPs that cannot be confined within a domain. An additional, and important, goal is that our path computation scheme should not violate the inter-domain confidentiality requirements mentioned in section I and should remain scalable. We assume that ingress has only local domain visibility which implies that it does not have topological or resource information other than its own domain. In summary, we need to find scalable, optimal or near-optimal inter-domain TE paths subject to confidentiality constraints.

VII. Simulation Experiments

We carried out our simulations for comparing CWS with the existing per-domain path computation scheme of [1]. In this paper, we report results for two inter-domain networks:

Network 1: Network 1 consists of 20 POPs and 46 bidirectional inter-domain links, each of capacity 800 units. It is a homogenous network topology which represents the Delaunay triangulation for the twenty largest metros in the continental US. Each POP node is considered a separate domain and we used an intra-domain 15-node, 28-link topology given in [13] at each node (POP) of the backbone network. Intra-domain links were chosen to have a capacity uniformly distributed between 100 and 300 units

Network 2: Network 2 uses the COST266 topology which consists of 28 domains and 37 bidirectional inter-domain links. Each inter-domain link was assigned a capacity of 500 units. At each of the 28 POP nodes, a 15-node random intra-domain network was generated in which each node pair is connected with 0.5 probability. Each intra-domain link was assigned a capacity uniformly distributed between 150 and 300 units.

For both networks, one of the edge nodes (DBRs) of the intra-domain topology was randomly chosen to establish an inter-domain link with another randomly chosen DBR of similar topology.

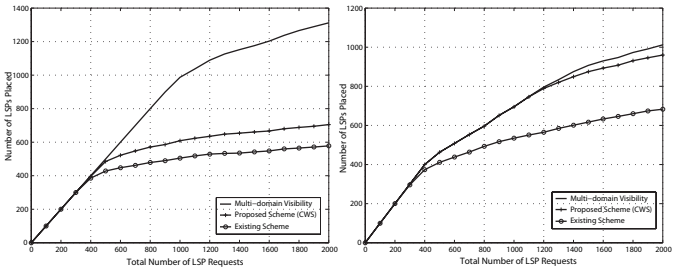

Fig. 1. Number of LSPs placed: Network 1 (left) and Network 2 (right).

The traffic matrix consists of a series of 2000 LSP requests along with their requisite bandwidth. For each LSP request, ingress and egress nodes were randomly chosen amongst all the nodes in the complete network. The bandwidth demand for each LSP request was uniformly distributed between 5 and 25 units with an infinite call holding time. This holding time is justified since we are considering networks deployed by large service providers where, typically, long-duration LSPs are established between large enterprizes or other service providers. LSP requests arrive one-by-one and if a bandwidth guaranteed path is found to place the LSP, the LSP request is accepted; otherwise, it is rejected and network is restored to the state it was in before the arrival of that LSP. As mentioned in section VI, an intra-domain TE LSP may be set up, in the case both ingress and egress are in the same domain which has sufficient TE resources (bandwidth, in this case) to satisfy the LSP request within the domain. Otherwise, an inter-domain route has to be found.

The performance of both per-domain path computation schemes that we evaluated was measured in terms of the average number of domains crossed per LSP, and the total number of LSPs that could be placed on a given network from a randomly generated traffic matrix. The results shown in figure 1 represent an average taken from 10 experiments for each of which a new traffic matrix was generated. The results show that for both networks used in simulations, the CWS scheme places many more LSPs as compared to the scheme in [1]. Furthermore, we also noticed (not illustrated in figure) that, for Network 1, each LSP traverses 18% fewer domains on average when CWS is used as compared to the number of domains traversed when the per-domain path computation scheme of [1] is used. Similarly, for Network 2, use of CWS resulted in an average of 20% fewer domains crossed by each LSP. Experiments performed on Polish POP topology resulted in similar improvements.

VIII. Conclusions

We presented a new inter-domain path computation scheme called CWS which tries to compute a near-optimal path without assuming availability of complete topology information. The CWS scheme conforms to practical constraints of routing domains whereby the service providers do not leak routing information outside the domain for confidentiality and scalability reasons. The CWS scheme was shown to work with simple addition of attributes to already existing signaling procedures.

The CWS scheme inherently guarantees that the computed inter-domain paths will traverse a minimum number of routing domains. A procedure to select a path among a set of candidate paths was also provided which ensures that the domain information remains confidential. We noticed that the path setup latency of CWS remains comparable to that of existing schemes; use of a new optional procedure within CWS further reduces the path setup latency.

We showed by simulations that CWS inter-domain path computation scheme exhibits significant improvement in terms of number of total routing domains traversed by an LSP. Consequently, fewer resources are utilized in the network resulting in placement of larger traffic demands. Finally, while the CWS scheme searches for better paths, it allows data flow as soon as the first feasible path is found, bringing the path setup latency close to that exhibited by schemes that stop after choosing the first available inter-domain path.

References

[1] J.-P. Vasseur, A. Ayyangar, and R. Zhang, "A Per-domain path computation method for establishing Inter-domain Traffic Engineering (TE) Label Switched Paths (LSPs)," Internet Draft, work in progress, August 2006.

[2] D. Awduche, L. Berger, D. Gan, T. Li, V. Srinivasan, and G. Swallow, "RSVP-TE: Extensions to RSVP for LSP Tunnels," RFC 3209, December 2001.

[3] R. Zhang and J.-P. Vasseur, "MPLS Inter-Autonomous System (AS) Traffic Engineering (TE) Requirements," RFC 4226, November 2005.

[4] J.-L. L. Roux, J.-P. Vasseur, and J. Boyle, "Requirements for Inter-Area MPLS Traffic Engineering," RFC 4105, June 2005.

[5] A. Farrel, J.-P. Vasseur, and A. Ayyangar, "A Framework for Inter-Domain MPLS Traffic Engineering," RFC 4726, November 2006.

[6] A. Farrel, A. Satyanarayana, A. Iwata, and G. R. Ash, "Crankback Signaling Extensions for MPLS and GMPLS RSVP-TE," Internet Draft, work in progress, January 2007.

[7] L. Berger, "Generalized Multi-Protocol Label Switching (GMPLS) Signaling Resource ReserVation Protocol-Traffic Engineering (RSVP-TE) Extensions," RFC 3473, January 2003.

[8] A. Farrel, D. Papadimitriou, J.-P. Vasseur, and A. Ayyangar, "Encoding of Attributes for Multiprotocol Label Switching (MPLS) Label Switched Path (LSP) Establishment Using Resource ReserVation Protocol-Traffic Engineering (RSVP-TE)," RFC 4420, February 2006.

[9] A. Farrel, J.-P. Vasseur, and J. Ash, "A Path Computation Element (PCE) Based Architecture," RFC 4655, August 2006.

[10] J.-P. Vasseur and J.-L. Le Roux (Editors), "Path Computation Element (PCE) communication Protocol (PCEP) – Version 1," Internet Draft, work in progress, January 2007.

[11] A. Ayyangar and J.-P. Vasseur, "Label Switched Path Stitching with Generalized MPLS Traffic Engineering," Internet Draft, work in progress, December 2006.

[12] K. Kompella and Y. Rekhter, "Label Switched Paths (LSP) Hierarchy with Generalized Multi-Protocol Label Switching (GMPLS) Traffic Engineering (TE)," RFC 4206, October 2005.

[13] S. Raza, F. Aslam, and Z. A. Uzmi, "Online Routing of Bandwidth Guaranteed Paths with Local Restoration using Optimized Aggregate Usage Information," in *Proceedings of the 2005 IEEE International Conference on Communications (ICC)*, Seoul, Korea, May 2005, pp. 201–207.