Hybrid Inter-Domain QoS Routing based on Look-Ahead Information

Ahmed Frikha, Samer Lahoud

Publications Internes de l'IRISA
ISSN : 2102-6327
PI 1946 – Mars 2010

Hybrid Inter-Domain QoS Routing based on Look-Ahead Information

AHMED FRIKHA[*] , SAMER LAHOUD[**]
ahmed.frikha@irisa.fr, samer.lahoud@irisa.fr

Abstract: Enabling the inter-domain routing with end-to-end Quality of Service (QoS) guarantees, remains a challenge for the next generation Internet. Most of the existing research in QoS routing has been done considering a single domain. However, extending the QoS routing to the inter-domain level faces two major challenges: the scalability and the domain autonomy. In this paper, we propose a novel inter-domain QoS routing algorithm based on a hybrid computation scheme, named HID-MCP. The hybrid computation scheme combines the on-demand computation scheme and the pre-computation schemes taking benefits from these two computation schemes. Particularly, the hybrid computation allows speeding up the response time while providing a high success rate. Moreover, our algorithm reinforces the domain autonomy and solves the scaling problem by distributing the computations between domains. Extensive simulations confirm the efficiency of our algorithm in terms of the success rate and the computational complexity.

[*] IRISA:ATNET - Universit Rennes 1
[**] IRISA:ATNET - Universit Rennes 1

1 Introduction

Nowadays, diverse advanced applications are proposed by the IP-based networks (e.g. IPTV, video-on-demand, telemedicine and e-health). Guaranteeing the Quality of Service (QoS) to such applications remains a difficult problem, especially when the service delivery requires crossing heterogeneous domains under the responsibility of different operators. In such case, the problem becomes more complex. Moreover, operators adopt different policies which consolidate their economic interests and confidentiality clauses. Thus, the cooperation between operators for providing QoS is realizable only if it respects their policies. Precisely, routing is one of the primary mechanisms for providing QoS. It consists in the computation of an end-to-end path which ensures the delivery of the service while meeting the QoS constraints. There are three different schemes to compute such path. The first scheme, named on-demand computation, computes the path upon the reception of the QoS request. The second scheme, named pre-computation, prepares in advance several paths or segments of paths which satisfy pre-determined QoS requests. These paths will be used at the reception of the QoS request to compute the end-to-end path. The third scheme is a hybrid one. It combines the pre-computation with the on demand computation. At the reception of QoS request, if the pre-computed paths do not satisfy the constraints imposed by the request, the hybrid alternative performs an on-demand computation.

In this paper, we investigate the inter-domain QoS routing problem based on hybrid compuation scheme. We propose a novel hybrid inter-domain QoS routing algorithm, named HID-MCP.

This paper is organized as follows. In section 2, a detailed description of the inter-domain QoS routing problem is presented as well as some related definitions. Section 3 investigates the different computation schemes for the QoS routing. Section 4 presents the concept of the HID-MCP algorithm and its operations. Simulation results are presented in detail in section 5 and a conclusion is given in section 6.

2 The Inter-Domain QoS Routing Problem

2.1 The QoS Routing Problem

The QoS routing, also called multi-constraint routing, consists in computing a path subject to multiple quality of service constraints between a source and a destination node of a network. Computing this path requires the knowledge of the QoS metrics on the network links. The QoS metrics can be classified into three types: bottleneck metrics such as bandwidth, additive metrics such as delay, and multiplicative metrics such as loss rate. The multiplicative metrics can be translated into additive metrics using the logarithm function. The bottleneck metrics can be resolved by omitting all links which violate the constraints and then computing the path on the residual graph. Therefore, we consider in the following only additive metrics.

Before formally defining the problem, we first introduce some notations. Let $G(N, E)$ denote a network topology graph, where N is the set of nodes (routers) and E the set of the links. Let m be the number of constraints. An m-dimensional weight vector is associated with each link $e \in E$. This vector consists of m non-negative QoS weights $w_i(e)$, $i = 1..m$. Let p be a path in the graph $G(N, E)$ and $w_i(p)$ be the weight of p corresponding to the additive metric i. Thus, $w_i(p)$ is given by the sum of the i^{th} weights of its component links: $w_i(p) = \sum_{e_j \in p}(w_i(e_j))$. Let $\overrightarrow{W}(p) = (w_1(p), w_2(p), ..., w_m(p))$ denote the weight vector of the path p.

Definition 1: The MCP problem

Given a source node s and a destination node d and a set of constraints given by the constraint vector $\overrightarrow{C} = (c_1, c_2, .., c_m)$, the Multi-Constraint Path (MCP) computation problem consists in finding a path p which satisfies $w_i(p) \leq c_i$, $\forall i \in 1..m$. Such path p is called a feasible path. The MCP problem may have zero, one or multiple solutions. Considering the set of feasible paths, we can define the non-dominated paths as follows: A path p is called non-dominated if there does not exist a path p' for which $w_i(p') \leq w_i(p)$ for all link weight components i except for at least one j for which $w_j(p') \prec w_j(p)$. The non-dominated paths can be classified using an energy function. The energy function associates a scalar cost for each weight vector. This classification leads to the following problem:

Definition 2: The MCOP problem

Considering an energy function F, a source node s, a destination node d and a constraint vector $\overrightarrow{C} = (c_1, c_2, .., c_m)$, the Multi-Constrained Optimal Path (MCOP) computation problem consists in finding a path p which satisfy:

1. p is feasible
2. $\forall p'$ a feasible path, $F(p') \geq F(p)$, where $F(p)$ is the scalar cost of p according to F.

Such path p is called optimal path. According to [1], a dominated path cannot be an optimal path. Hence, in the following, the dominated paths are discarded from the computation search space of the QoS routing algorithms.

In the literature, many algorithms are proposed to solve the MCP problem. Exact solutions such as the SAMCRA algorithm [2] and the Depth First Search (DFS) approach [3], can always solve the MCP problem if a feasible solution exists. Moreover, SAMCRA solves the MCOP problem. It always returns the shortest path according to the non-linear energy function $\max_{i\in 1..m}(\frac{w_i(p)}{c_i})$ as it stores at each node all non-dominated paths. Heuristic algorithms such as TAMCRA [4] and H-MCOP [5] algorithms are also proposed in order to solve the MCP and MCOP problems. These algorithms find feasible solutions with high probability while reducing the computation complexity. First, the H-MCOP heuristic is destined to solve MCOP problem. It executes a modified versions of the Dijkstra's algorithm twice: in forward direction using a linear energy function and in backward direction using a non-linear energy function. Second, TAMCRA is a heuristic of the exact algorithm SAMCRA. It limits the maximum number of paths stored at each node. Thus, TAMCRA does not always find a solution to the MCP problem. However, it is more suitable than SAMCRA for large scale networks as its computational complexity is lower.

2.2 The Inter-domain Routing Problem

A domain is a set of interconnected nodes forming an autonomous system (AS). Each domain is interconnected with other domains through its border nodes (BN), these nodes allow inter-domain communications. Each domain is under the responsibility of an operator. In practice, cooperation between different operators is limited. As the operators can be in competition, information about the internal topology or the available resources in the network is confidential. Another important issue related to the inter-domain context is the scaling problem. This issue has a direct impact on the routing problem. In fact, computing the path which satisfies the constraints through a sequence of domains is more complex because of the great number of nodes involved in the computation. Currently, the inter-domain routing protocol corresponds to BGP. This protocol does not take into account the QoS constraints. Many extensions of BGP are proposed to support the QoS routing [6]-[7]. However, the QoS capabilities of these propositions remain limited [8]. Recently, the research community is exploring the use of distributed solutions to solve the inter-domain QoS routing problem, such as the BRPC technique [11] based on the usage of computation entities called PCEs (Path Computation Elements).

3 Computation Schemes for QoS Routing

Three computation schemes are proposed to solve the QoS routing problem: the *on-demand computation* scheme, the *pre-computation* scheme and the *hybrid computation* scheme. The on-demand computation scheme seeks to find a feasible path for each request based on the network state information. The computation is triggered upon the reception of the QoS request. This computation scheme provides a high probability of finding a feasible path since the network state information is instantaneous. This scheme is largely used in the networks but presents some serious limitations with the emerging applications in the Internet. In fact, the MCP problem is NP-hard, consequently, the performance of the on-demand routing in terms of response time are severely affected.

In contrast with the on-demand computation scheme, the pre-computation scheme enables to solve the QoS routing problem while speeding up the response time. The pre-computation proceeds in two phases: It prepares in advance a set paths satisfying predetermined QoS requests in a first phase. Then, at the reception a of QoS request, it seeks to rapidly provide a feasible path among the pre-computed paths. Ideally, the second phase needs only to select one of the pre-computed solutions. However, some additional computations may be performed. For instance, when handling QoS requests with delay constraints, the first phase may pre-compute feasible paths for a wide range of possible delay constraints, while the second phase just needs to select a suitable path from the pre-computed set, *i.e.*, one that satisfies the particular delay constraints of the request. The execution time of the second phase has an immediate impact on network performance; hence, it is highly desirable to keep its computational complexity as low as possible. In the above example, the less time is consumed in finding the proper path, the less time is spent in establishing the new request.

However, the deployment of a pre-computation scheme in the networks can run into problems. Precisely, the stored paths are computed based on a snapshot of the network state taken before the reception of the QoS request. Consequently, the pre-computed paths do not necessarily satisfy the constraints imposed by the QoS constraints since the taken snapshot is not valid after an eventual change in the network state. Hence, this scheme requires the update of network state information periodically or using a network state-dependent threshold. In addition, a pre-computation scheme cannot forecast all possible QoS request. Therefore, even if the snapshot is valid, the pre-computed paths do not necessarily meet all feasible QoS requests. This decreases the success rate in the pre-computation scheme.

Consequently, a hybrid computation scheme is proposed. This alternative combines the two aforementioned computation schemes and is performed in two phases. As the pre-computation scheme, the first phase consists in preparing in advance a set of paths or segment of paths. The second phase is more sophisticated. When the pre-computed paths or segment of paths do not lead to find a path which satisfies the request, the on-demand computation is triggered. Therefore, the hybrid computation scheme ensures a high acceptance rate of the requests while reducing the computation time. In the following,

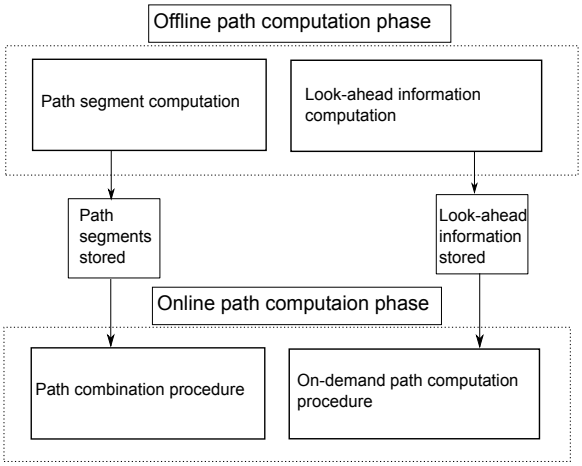

Figure 1: The building bock architechure of the HID-MCP algorithm

we detail the benefits of combining the on-demand and the pre-computation scheme in a hybrid scheme.
Acceptance rate. The hybrid computation scheme ensures a high acceptance rate of the QoS requests since it allows the on-demand computation when the pre-computed paths cannot satisfy the QoS request.
Response time. Thanks to the pre-computation phase of the hybrid computation scheme, some QoS requests are accepted without executing the on-demand computation. Furthermore, infeasible QoS requests can be filtered and rapidly rejected using the look-ahead information deduced from the pre-computed paths. This decreases considerably the response time. Other requests necessitate an on-demand computation, however the response time of this phase is speeded up significantly by reducing the search space with the help of the look-ahead information.

In addition to the aforementioned benefits, the hybrid computation scheme profits from the specific advantages of the pre-computation scheme. We summarize these advantages in the following.
Traffic Engineering. One major advantage of the path pre-computation is the enhancement of the traffic engineering capabilities of QoS routing. Particularly, the requests can be efficiently routed in order to globally optimize the network resources while satisfying the QoS constraints. For instance, a pre-computation may supply different paths to different QoS requests according to their class of service. This enables an efficient load balancing and thus optimizes the usage of network resources.
Scalability. QoS routing must scale well with the growth of the network size and the augmentation of the number of requests. The hybrid computation scheme enhances the scalability of QoS routing considering the two aforementioned dimensions. Particularly, a pre-computation phase reduces the computational load on network elements; when the request arrival rate is high, pre-computation allows the reduction of the overall computation load by reusing pre-computed paths. Moreover, pre-computation uses the concept of class of services in order to efficiently respond to requests with diverse QoS constraints. Pre-computation also enables using the concept of topology aggregation, thus limits the amount of link state information and as a result limits the scalability concerns.
Dependability. When the applications are mission critical, the dependability of the network becomes an important factor. Pre-computation enhances the dependability of the network by reducing the response time to failures; this can be done by pre-computing the backup paths. When the failure occurs, the traffic is rapidly switched from the nominal path to the pre-computed backup path. In highly dependable networks, backup paths may be pre-computed for each possible network element failure.

4 The Proposed Algorithm: HID-MCP

In this paper, we propose a novel inter-domain QoS routing algorithm, named HID-MCP (Hybrid Inter-Domain MCP). HID-MCP is based on a hybrid computation scheme. The present section describes this algorithm.

4.1 Concept

In this section, we detail the operations performed by the HID-MCP algorithm. Figure 1 illustrates the building bock architecture of the algorithm. This architecture is implemented in each domain. Precisely, based on this architecture, each domain performs some computations and cooperates with other domains to compute the end-to-end path. This reinforces the domain autonomy and confidentiality. As shown in this figure, the HID-MCP algorithm consists of two phases. In the

first phase, named offline path computation phase, the algorithm executes an intra-domain pre-computation algorithm and computes *look-ahead* information. The pre-computed paths and the look-ahead information are stored in a database for later use.

The second phase, named online path computation phase, is triggered upon the reception of a QoS request. In this phase, HID-MCP computes an end-to-end path that spans multiple domains and fulfils the QoS constraints. This phase includes two procedures. The first procedure, named path combination procedure, seeks to find a feasible path by combining the pre-computed paths stored in each domain. When the combination procedure fails to find a feasible path, a second procedure is called. This procedure executes an improved on-demand computation algorithm in order to find a feasible path. This procedure benefits from the stored *look-ahead* information to maintain a low computational complexity. In the following, we explain the operations performed in the two aforementioned phases of the HID-MCP algorithm:

4.1.1 The Offline Path Computation Phase

The offline computation phase consists in computing in advance a set of intra-domain paths subject to multiple predetermined QoS constraints. It also computes *look-ahead* information at the level of each entry border node of the corresponding domain. In the following, we detail the operations involved in these two computations.

The Path Segment Computation The path segment computation procedure seeks to pre-compute a set of paths that link the entry border nodes of the domain to the any node in N, where N is the set of nodes of the domain union the set of entry border nodes of the neighboring domains. These paths are further used by the path combination procedure of the online path computation phase. Based on the study of the pre-compuation algorithms carried in [9], we perform the intra-domain pre-computations using the *ID-PPPA* algorithm [9]. *ID-PPPA* is based on a simple single-weight computation of shortest path trees rooted at the border nodes. Each shortest path tree, also called a primary path tree, is computed using the Dijkstra algorithm and considering a single weight w_i for each link i. Therefore, *ID-PPPA* computes the m shortest path trees that minimise respectively a single QoS metric. In [9], we showed that *ID-PPPA* is a fast heuristic for the MCP problem. *ID-PPPA* necessitates the computation of m shortest path trees per border node, where m is the number of metrics. The complexity of *ID-PPPA* depends on the number of constraints m. For one border node, *ID-PPPA* is in $\mathcal{O}\left(m(N\log(N)+E)\right)$ corresponding to m times the complexity of the Dijkstra algorithm. The global complexity is then given by: $\mathcal{O}(Bm(N\log(N)+E))$, where B is the number of the entry border node of the domain.

Look-Ahead Information Computation During the offline phase of HID-MCP, we propose to compute *look-ahead* information in each domain. This information gives a measure of the best QoS performance that can be provided by the domain. Particularly, it enables to reduce the computation search space of the on-demand path computation procedure. For instance, this information enables to discard non feasible paths from the search space of the procedure before exploring these paths. Therefore, *look-ahead* information reduces the computational complexity of the online phase and contributes in maintaining a reasonable response time. *Look-ahead* information is inferred from the result of the pre-computation algorithm. Let us denote by $P^*_{n_1\mapsto n_2;i}$ the pre-computed shortest path between border node n_1 and node n_2 considering the metric i. Then, $w_i(P^*_{n_1\mapsto n_2;i})$ gives the lowest possible path weight between n_1 and n_2 considering the metric i. Similarly, let us denote by $\overrightarrow{b}_{n_1\mapsto n_2}=(b_1,..,b_m)$ the vector where $b_i = w_i(P^*_{n_1\mapsto n_2;i})$. $\overrightarrow{b}_{n_1\mapsto n_2}$ represents the lowest path weight to reach n_2 from n_1. We note that this vector does not correspond necessarily to an existing path. However, this vector can be used in the online path computation phase to discard non feasible paths from the search space. For example, let $\overrightarrow{b}_{n_1\mapsto n_2}=(7,5)$ be the *look-ahead* information to reach n_1 from n_2, where $m=2$. Any request to reach n_1 from n_2 which has at least one constraint lower than the corresponding metric in $\overrightarrow{b}_{n_1\mapsto n_2}$ is not feasible. For instance, the request $\overrightarrow{C}_{n_1\mapsto n_2}=(6,5)$ is not feasible.

Theorem 1: The complexity of the look-ahead information computation
The complexity of the *look-ahead* information computation is given by : $\mathcal{O}(mNB)$.

Proof: The *look-ahead* information is inferred from the result of the path segment computation. At each entry border node of the domain, there are at most mN stored pre-computed paths. Hence, at the level of an entry border node n the complexity of computing the N vectors $\overrightarrow{b}_{n\mapsto n_j}$, where $n_j \in N$, is in: $\mathcal{O}(mN)$. Therefore, the complexity of computing the *look-ahead* information for all the entry border nodes of the domain is in: $\mathcal{O}(mNB)$.

4.1.2 The Online Path Computation Phase

The online path computation procedure consists in finding a feasible end-to-end path using the pre-computed paths and taking benefit of the *look-ahead* information. Upon the reception of a QoS request, the source and the destination are determined. According to the cooperation policy, the service provider computes the best shortest domain sequence that link the source and the destination []. The pre-computed path combination procedure is triggered in the desination domain toward the source domain following the computed domain sequence.

The Pre-computed Path Combination Procedure This procedure seeks to compute an inter-domain path by combining the paths pre-computed at the offline phase. At the reception of the QoS request, the procedure is launched at the level of the destination domain. The proposed algorithm selects the feasible paths that lead to the up-stream domain. These paths are sent to the up-stream domain using a novel compact structure named VSPH (Virtual Shortest Path Hierarchy). This structure contains only the extremity of the paths (destination node and the entry border node of the up-stream domain) and the weight vector of each path. This structure allows preserving the confidentiality of the domains. When receiving the VSPH, an intermediate domain combines the paths in the VSPH with the internally pre-computed paths. Then, it selects the feasible paths, computes the a VSPH and sends it to the up-stream domain.

This procedure is recursive: each domain performs the same operations and this leads to an end-to-end path. However in some cases, the combination procedure is interrupted because no feasible path is found. In this case, two different approaches are proposed in order to overcome this limitation. The first approach of HID-MCP is based on local improvement while the second approach is based on cranckback improvement. These approaches are explained in the following.

Theorem 2: The complexity of the pre-computed path combination procedure
The complexity of the pre-computed path combination procedure at the level of an intermediate domain is in: $\mathcal{O}(m^2 B_{max}^2)$, where B_{max} denotes the maximum number of border nodes between two domains.

Proof The pre-computed path combination adds at each entry border node the weight of the stored pre-computed paths and the weight of the paths in the VSPH. There are at most m paths from the destination to each entry border node of the domain. In addition, at each entry border node, there are at most mB_{max} stored pre-computed paths to reach the upstream domain. Hence, the complexity of combining the pre-computed paths and the received paths at the level of an entry border node is in: $\mathcal{O}(m^2 B_{max})$. This operation is performed at each entry border node between the domain and the downstream domain. Therefore, the global complexity of this procedure at each domain is in: $\mathcal{O}(m^2 B_{max}^2)$.

HID-MCP based on Local Improvement Figure 2 illustrates the operations of the HID-MCP based on local improvement. This approach suspends the pre-computed path combination procedure and executes the on-demand path computation procedure in the current domain (i.e. where the combination is stopped). The on-demand procedure is explained in the following section. Then, if a feasible path is found in the current domain this path is sent to the up-stream domain which will continue the pre-computed path combination procedure. Otherwise, if the algorithm does not find a solution in the current domain the request is rejected.

HID-MCP based on Cranckback Improvement The HID-MCP based on cranckback improvement is explained in figure 3. This approach stops the pre-computed path combination procedure and sends a cranckback message to the destination domain to execute the on-demand path computation procedure beginning from the destination domain. Each domain executes the on-demand path computation procedure and sends the computed VSPH to the up-stream domain. The computation stops when an end-to-end path is found or when the on-demand path computation procedure does not find a solution.

The On-demand Path Computation Procedure When the pre-computed path combination procedure does not lead to find a feasible path, the on-demand path computation procedure is called in the current domain or starting from the destination domain according to the two aforementioned approaches.

The goal of this procedure is to provide paths more adapted to the request than the pre-computed paths. According to our study in [9], we propose the use of the TAMCRA algorithm to perform the on-demand computation. However, we integrate some modifications into this algorithm. This version takes benefit of the *Look-Ahead* information computed at the offline path computation phase. This information allows discarding the non-feasible paths from the received VSPH before executing TAMCRA.

Before explaining the operations performed of by the on-demand procedure, we first introduce some notations. Let P be a path in the VSPH from the destination d to the entry border node n of the current domain, and $\overrightarrow{W}(P) = (w_1(P), w_2(P), ..., w_m(P))$ be its weight vector. We define the weight vector $P_{d \mapsto n_j} = \overrightarrow{W}(P) + \overrightarrow{b}_{n \mapsto n_j}$ for each $n_j \in B_1$, where B_1 is the set of the entry border node of the up-stream domain. Note that B_1 corresponds to the source node if it is within the current domain. The the weight vector $P_{d \mapsto n_j}$ represents the sum of the weight vector of the crossed path P and the lowest weight vector to reach n_j from n. Thus we can define a new cost for P given by: $C(P) = \min_{n_j \in B_1} \left\{ \max_{i \in 1..m} \left(\frac{w_i(P_{D \mapsto n_j})}{l_i} \right) \right\}$. A path P, that has a cost $C(P) > 1$, is infeasible since it cannot lead to any node in B_1 while meeting the QoS constraints. Hence, it is discarded from the VSPH. In addition, we define a new parameter, denoted by l, for this procedure. In fact, we select the l shortest feasible paths from the VSPH using the cost C. Then, we initialize the border nodes of the l shortest path by the corresponding weight vectors in the VSPH. Finally, the procedure executes the TAMCRA algorithm at each initialized nodes to reach the entry border node of the up-stream domain or the source node. This reduces considerably the computation complexity of the procedure and decreases the number of paths exchanged between domains. We note that at

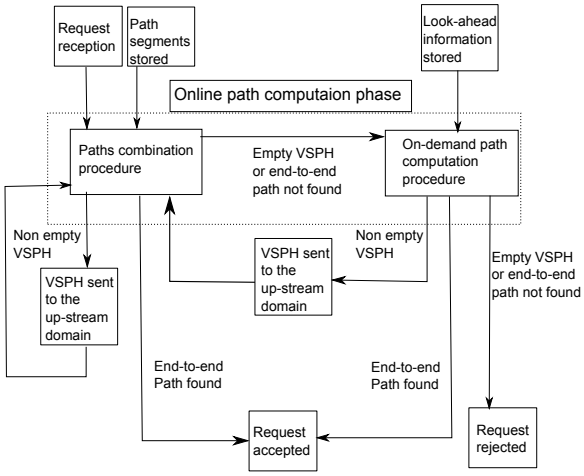

Figure 2: HID-MCP based on local improvement

the destination domain, there is no received VSPH. Hence at the level of the destination domain, the on-demand procedure executes the TAMCRA algorithm at the destination to reach the entry border node of the up-stream domain or the source node.

Theorem 3: The complexity of the on-demand path computation procedure
The complexity of the on-demand path computation procedure at the level of an intermediate domain is in: $\mathcal{O}(l(kN\log(kN)+k^3mE))$ corresponding to l times the complexity of the TAMCRA algorithm.

Proof: The on-demand path computation procedure relies on three phases: (1) adding the weights of the paths in the VSPH to the corresponding *look-ahead* information, (2) filtering the infeasible paths, selecting the l shortest paths and initializing the correspondent entry nodes and (3) executing the TAMCRA algorithm at each initialized border nodes. There are at most α paths from the destination to each entry border node of the domain, where $\alpha = m$ in the HID-MCP based on local improvement and $\alpha = k$ in the HID-MCP based on crankback improvement. Note that k corresponds to the maximum number of stored paths paths at each intermediate node in the TAMCRA algorithm. In addition, at each entry border node, there are at most B_{max} stored look-ahead information to reach the upstream domain. Hence, the complexity of the operation (1) is in: $\mathcal{O}(\alpha B^2_{max})$. Filtering the infeasible paths and selecting the l shortest path necessitate the comparison of the αB^2_{max} weights computed in the operation (1). Therefore, the complexity of the operation (2) is in: $\mathcal{O}(\alpha^2 B^4_{max})$. Finally, the algorithm executes the TAMCRA algorithm at each initialized node. Knowing that the number of initialized node is less or equal to l, the complexity of the operation (3) is in: $\mathcal{O}(l(kN\log(kN)+k^3mE))$.

The value of α is usually small (below10), *i.e.* the number of metrics $m \leq 10$ and the parameter of TAMCRA $k \leq 10$. Consequently, the critical point that determines the complexity of the on-demand path computation procedure is the number of node N and the number of border nodes between two domains B_{max}. Assuming that $B_{max} < N\log(N) + E$ [12], the complexity of this procedure is in : $\mathcal{O}(l(kN\log(kN)+k^3mE))$.

4.2 Example

In this section we explain the operations performed by the HID-MCP algorithm based on a simple example of three domains. In this example, the number of constraints m equals 2.

Figure 4 illustrates the operations performed by the pre-computed path combination procedure. Let $\vec{C}_{S\rightarrow D}= (10, 10)$ be the received QoS request, where S is the source node and D is the destination node. The service provider computes the domain sequence to be crossed. At the level of the destination domain (Domain 1), the pre-computed paths from D to the entry border node of the Domain 2 $\{V_3, V_4\}$ are selected. The paths $\{D, V_1, V_2, V_3\}$ and $\{D, V_1, V_2, V_4\}$ minimize the first metric, and the paths $\{D, V_2, V_3\}$ and $\{D, V_2, V_4\}$ minimize the second metric. The VSPH 1 is computed and sent to the Domain 2. At the level of the Domain 2, the pre-computed paths from $\{V_3, V_4\}$ that lead to the Domain 3 are combined with the received paths in the VSPH 1. Infeasible paths are discarded. For instance, the combination of the pre-computed path $\{V_4, V_3, V_5, V_6, V_7\}$ and the path with the weight vector equal to $(5, 3)$ in the VSPH 1 leads to the infeasible path $\{D, V_4, V_3, V_5, V_6, V_7\}$ with the weight vector equal to $(12, 7)$. The VSPH 2 is sent to the Domain 3. This domain combines the pre-computed paths that lead to the source and the received paths. Unfortunately, no feasible path is found. Here, the local improvement-based HID-MCP executes the on-demand path computation procedure in the Domain 3, while the crankback improvement-based HID-MCP sends a crankback message to the Domain 1 to execute the same procedure, but beginning from the Domain 1.

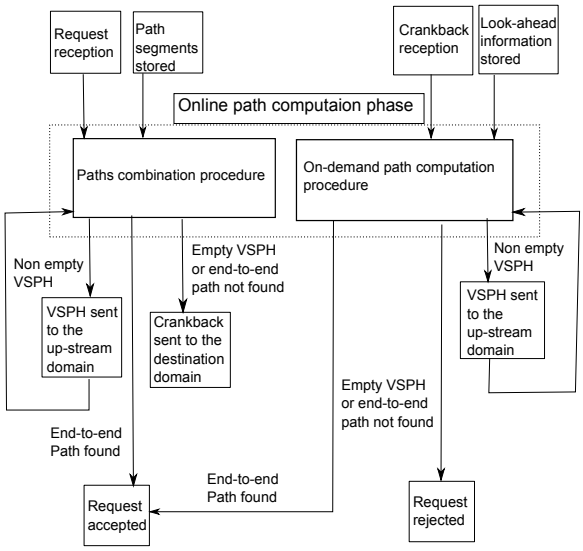

Figure 3: HID-MCP based on cranckback improvement

Figure 5 illustrates the operations performed of this procedure in the Domain 3. In this example, the parameter k of TAMCRA equals 2 and the parameter l (the number of shortest paths selected from the VSPH) is equals 2. The vector $\overrightarrow{b}_{V_7 \mapsto S} = (2,1)$ represents the lowest weight vector to reach S from V_7 while the vector $\overrightarrow{b}_{V_8 \mapsto S} = (3,3)$ represents the lowest vector cost to reach S from V_8. These Vectors are deduced respectively from the pre-computed paths: $\{\{V_7, V_9, S\}, \{V_7, S\}\}$ and $\{\{V_8, V_{10}, V_7, S\}, \{V_8, V_{10}, V_7, S\}\}$. $\overrightarrow{b}_{V_7 \mapsto S}$ and $\overrightarrow{b}_{V_8 \mapsto S}$ represent the *Look-Ahead* information computed at the offline path computation phase. The on-demand path computation procedure adds the *look-ahead* information at each entry border node of the Domain 3 to each received path in VSPH 2. Then, it filters the infeasible paths and selects the l shortest path according the cost C. After the infeasible path filtering from the VSPH 2 only the path with vector cost equals $(8,9)$ remains in the VSPH 2. Node V_7 is initialized by this cost and the TAMCRA algorithm is executed. In this example the procedure does not find a feasible path, the HID-MCP algorithm is stopped, and the request is rejected.

Figure 6 illustrates the operations performed by the HID-MCP after the crankback message reception. At the level of the destination domain, the on-demand path computation procedure begins. TAMCRA is executed in this domain at the destination node. Two paths are saved at each node since $k = 2$. Hence two paths are computed at each entry border node, the VSPH 1 sent to the Domain 2 contains four shortest paths. At the reception of the VSPH 1, Domain 2 computes the l shortest paths in the VSPH 2 using the *Look-Ahead* information. The two shortest paths correspond to the path with the weight vector $(4,4)$ and $(5,3)$. The border node which contains one or more shortest paths is initialized. Thus, V_3 and V_4 are initialized by the corresponding cost vector. The algorithm executes TAMCRA twice, one at the entry border node V_3 and the other at the entry border node V_4. The computed non dominated paths are sent under the VSPH 2 structure to the Domain 3. This domain selects only the paths with the cost equals $(7,8)$ since the other paths in the VSPH 2 are not feasible according to the *Look-Ahead* information. Finally, two feasible paths are returned after the execution of TAMCRA at the node V_7.

5 Simulation and analysis

In this section, we evaluate the performance of the hybrid QoS routing algorithm HID-MCP. The simulations are performed using a network of three domains where each domain is built based on Waxman's model with 50 nodes in each domain. The probability that two nodes of the network are connected by an edge is expressed in [10]. We associate with each link two additive weights generated independently following a uniform distribution [10,1023]. The QoS constraints are also randomly generated according to the following : Let p_1 and p_2 denote the two shortest paths which minimize the first and the second metric, respectively. Let $Z = [w_1(p_1), w_1(p_2)] \times [w_2(p_2), w_2(p_1)]$ be the constraint generation space. As shown in figure 7, we divide this space into 10 zones Z_i, $i = 1..10$ and we browse the space from the strictest constraint zone Z_1 to the loosest constraint zone Z_{10}. Then, we assess the perfomance of our algorithm according to these zones.

We evaluate our algorithms based on the following performance criteria:

- GSR: the global sucess rate given by the ratio between the number of the requests where a feasible path is found and the total number of QoS requests.

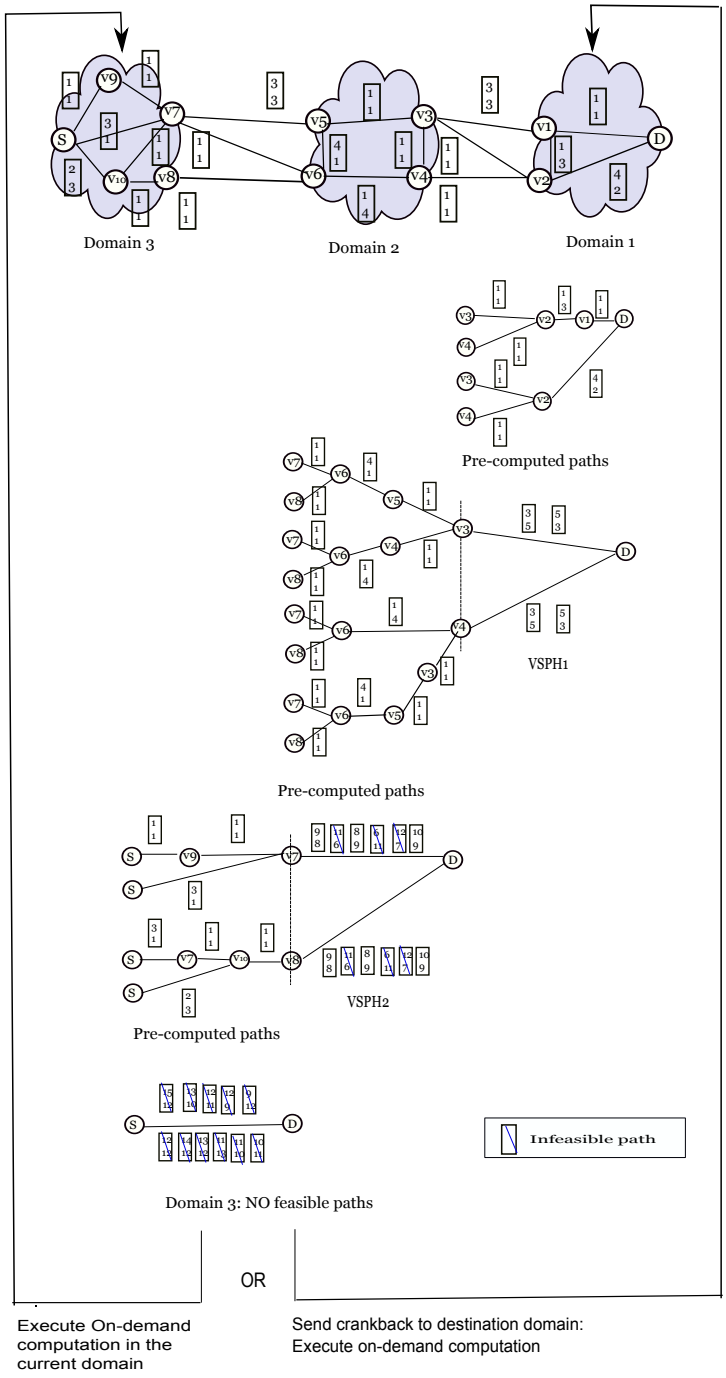

Figure 4: The combination procedure example for $m = 2$, $k = 2$ and $l = 2$

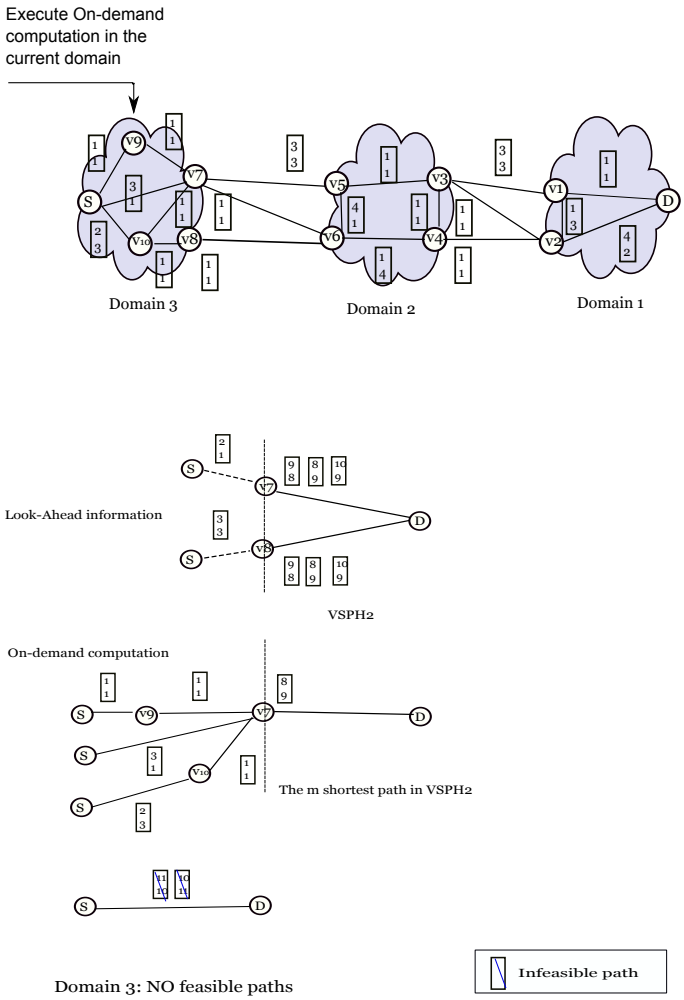

Figure 5: Example of the HID-MCP algorithm based on local improvement when $m = 2$, $k = 2$ and $l = 2$

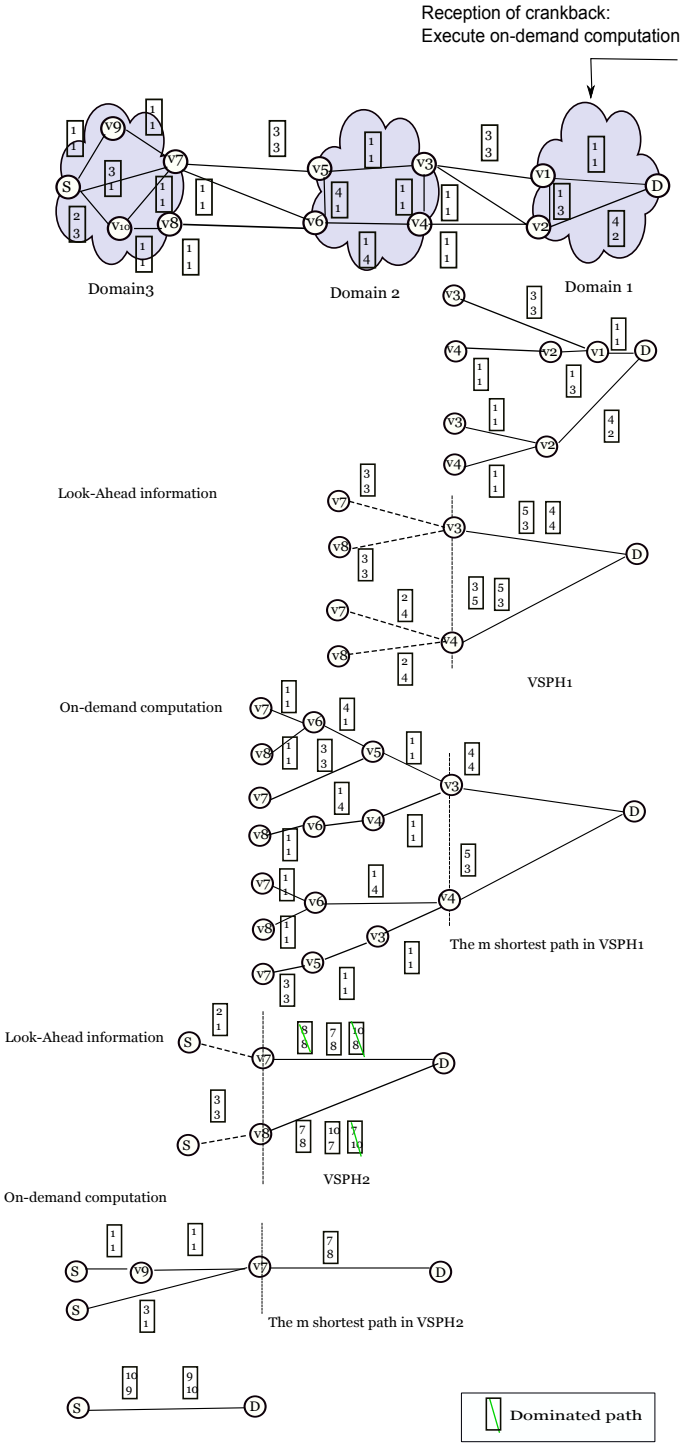

Figure 6: Example of the HID-MCP algorithm based on cranckback improvement when $m = 2$, $k = 2$ and $l = 2$

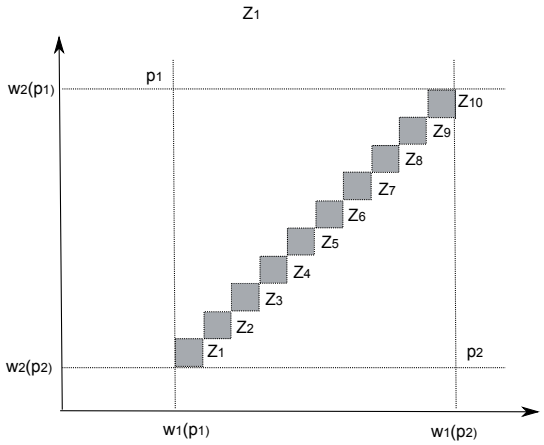

Figure 7: Constraint generation zones with $m = 2$

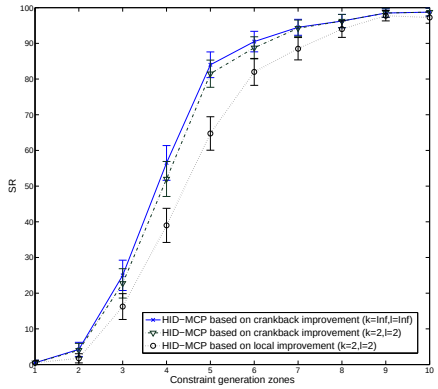

Figure 8: Success rate of the HID-MCP algorithm with the two approaches

- PIP: the percentage of the infeasible paths in the VSPH received by the domain i given by the ratio between the number of infeasible paths in the VSPH and the Number of paths in the VSPH.
- CSR: the efficiency of the combination procedure at domain i given by the ratio between the number of requests where the combination procedure is successful (e.g. leading to at least one feasible segment between the destination and the border nodes of upstream domain) and the number of received requests at domain i.

In the following, each figure contains multiple curves measuring the variation of one performance metric according to the strictness of quality service constraints. Each curve is represented by ten points. Each point i of a curve represents the average value of the performance metric obtained when constraints are generated in Z_i. Each average value is associated with a 95% confidence interval *i.e.* with a risk threshold equal to 5%.

Figure 8 illustrates the variation of the success rate (SR) of the two approaches of the HID-MCP algorithm according to the strictness of the QoS constraints. In this figure, we compare the success rate of the HID-MCP algorithm based on local improvement to the SR of the HID-MCP based on crankback improvement while varying the two parameters of the algorithm l and k. As explained in 4, k denotes the maximum number of stored path at each intermediate node when executing TAMCRA while l denotes the maximum number of paths selected from the VSPH. Note that HID-MCP based on crankback improvement is an exact algorithm when $l = \infty$ and $k = \infty$, *i.e* the algorithm solve the MCP problem if a feasible solution exists. The figure shows that the success rate of the HID-MCP based on crankback improvement when $l = 2$ and $k = 2$ is very close to the success rate of the exact algorithm. As expected, the success rate of the HID-MCP based on local improvement when $l = 2$ and $k = 2$ is lower than that of the HID-MCP based on crankback improvement with the same parameters. In fact, when the combination procedure fails, the crankback improvement-based HID-MCP executes the on-demand computation procedure beginning from the destination domain while the local improvement-based HID-MCP executes it only in the current domain. Consequently, the quality of the paths computed by the crankback improvement in each domain is better than that of the local improvement. This improves the probability to find a feasible path in the crankback improvement, but increases the computational complexity comparing with the local improvement.

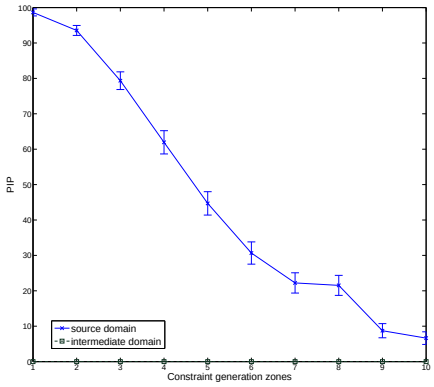

Figure 9: Percentage of infeasible paths in the received VSPH in each domain

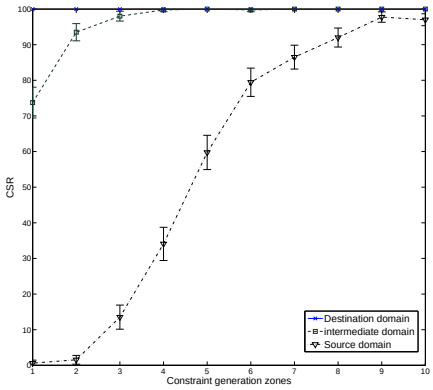

Figure 10: Success rate of the combination procedure in each domain

As explained in the section 4.1, the *look-ahead* information decreases the computational complexity of the on-demand computation procedure: first by discarding the infeasible paths from the received VSPH, second by classifying the paths in the VSPH to select only the l shortest paths. Figure 9 illustrates the percentage of the infeasible paths in the VSPH received in each domain (PIP). We note that this percentage equals 0 in the intermediate domain. In fact, the weights of the paths received from the destination domain are still low. Therefore, detecting infeasible paths using the *look-ahead* information in the intermediate domain is unlikely to happen. However, in the source domain the percentage of the discarded infeasible paths is high, especially when the QoS constraints are strict. This figure shows the importance of the *look-ahead* information in discarding infeasible paths especially in the source domain. Nonetheless, the *look-ahead* information remains important to select the l shortest path in all the crossed domains.

In the following, we focus on the success rate of the combination procedure (CSR) in a given domain i. This performance criteria measures the probability of the combination procedure success in the domain i. The complimentary of this probability, corresponds to the percentage of executing the on-demand computation procedure in this domain. Figure 10 illustrates the variation of the CSR in each domain according to strictness of the QoS constraints. In the destination domain, the combination procedure is always successful. In the intermediate domains the success rate of this procedure is high and equals 100% when constraints are not very strict. However, we note that the CSR in the source domain is low when the constraints are strict. Nonetheless, this procedure performs well when the constraints are less strict. From this figure, we deduce that the probability of executing the on-demand computation is high only in the source domain when the QoS constraints are very strict. This proves that the empirical complexity of HID-MCP remains reasonable.

6 Conclusion

In this paper, we studied the inter-domain QoS routing problem. We analyzed the existing computation schemes for the QoS routing. Particularly, we proposed a novel inter-domain QoS routing algorithm based on a hybrid computation scheme, named HID-MCP. The offline phase of HID-MCP consists in preparing in advance a set of QoS paths and computing *look-ahead* information. The online phase, triggered at the reception of QoS request, tries to combine the pre-computed path segments in order to obtain an end-to-end feasible path. In addition, we introduced two different approaches for improving the success

rate of the online phase. The first approach is based on local improvement, while the second approach is based on crankback improvement. Extensive simulations showed that both of the two aforementioned approaches provide a high success rate and maintain a low complexity time. Precisely, the crankback improvement-based HID-MCP has a high success rate very close to that of exact approach, while the local improvement-based HID-MCP provides lower computational complexity. This gives the choice to the operators to execute either the local improvement approach or the crankback improvement approach, depending on the computation policy and the priority of the request.

References

[1] Van Mieghem, P. and Kuipers, F. A. *Concepts of exact QoS routing algorithms*, IEEE/ACM Transactions on Networking, 2004, 12, 851-864

[2] Van Mieghem, P. , De Neve, H., and Kuipers, F. A. *Hop-by-hop quality of service routing*, Computer Networks, 2001, 37, 407-423

[3] Tarjan, R. E. *Depth-First Search and Linear Graph Algorithms*, SIAM Journal of Computers, 1972, 1, 146-160

[4] De Neve, H. and Van Mieghem, P. *TAMCRA: a tunable accuracy multiple constraints routing algorithm*, Computer Communications, 2000, 23, 667-679

[5] Korkmaz, T. and Krunz, M. *Multi-constrained optimal path selection*, Twentieth Annual Joint Conference of the IEEE Computer and Communications Societies (INFOCOM), 2001, 2, 834-843

[6] T. Knoll, *BGP Extended Community Attribute for QoS Marking*, draft-knoll-idr-qos-attribute-02, work in progress, IETF, 2009.

[7] D. Griffin, J. Spencer, J. Griem, M. Boucadair, P. Morand, M. Howarth, N. Wang, G. Pavlou, A. Asgari, and P. Georgatsos, *Interdomain routing through QoS-class planes*, IEEE Commun. Mag., vol. 45, no. 2, pp. 8895, Feb. 2007.

[8] R. Mahajan, D. Wetherall, and T. Anderson, *Towards coordinated interdomain traffic engineering*, in SIGCOMM Workshop on Hot Topics in Networking (HotNets). New York, NY, USA: ACM, 2004.

[9] Frikha, A., Bertrand, G., and Lahoud, S.*Pré-calcul de chemins inter-domaines soumis à plusieurs contraintes de qualité de service* Tech. Rep. 1935, IRISA(2009). http://hal.inria.fr/inria-00319401,ISSN: 2102-6327

[10] K.I. Calvert, M.B. Doar, and E.W. Zegura, *Modelling Internet Topology*, IEEE Communications Magazine, 35(6), pp.160-163, June 1997

[11] J.P. Vasseur, R. Zhang, N. Bitar, and JL. Le Roux. *A Backward-Recursive PCE-Based Computation (BRPC) Procedure to Compute Shortest Constrained Inter-Domain Traffic Engineering Label Switched Paths*, RFC 5441 IETF, April 2009.

[12] N. Spring, R. Mahajan, D. W., and Anderson., T. *Measuring ISP topologies with Rocketfuel* IEEE/ACM Transactions on Networking, 2004.