On Cooperative Inter-Domain Path Computation

Payam Torab, Bijan Jabbari, Qian Xu and Shujia Gong
George Mason University
{ptorab, bjabbari, qxu, sgong}@gmu.edu

Xi Yang and Tom Lehman
Information Sciences Institute East, University of Southern California
{xyang, tlehman}@isi.edu

Chris Tracy and Jerry Sobieski
Mid-Atlantic Crossroads, University of Maryland
{chris, jerrys}@maxgigapop.net

Abstract

Inter-domain path computation, or the ability to compute end-to-end paths across multiple domains, is the next step toward wide deployment of a distributed control plane with support for traffic engineering. A key enabler to achieve this goal is the introduction of a Path Computation Element (PCE) in each domain. There are various ways these elements can collaborate to compute an end-to-end path; of particular interest to us in this paper is cooperative path computation, a scheme where PCEs exchange path information in the context of a specific end-to-end path computation instance, often prior to signaling the path. We show that depending on the information available to each PCE, cooperation can take one of two forms, which we call model-based and ad hoc. We demonstrate that model-based cooperation is essentially a multistage decision problem, and offer a probabilistic analysis which we believe is the key to understanding the problem and developing efficient inter-domain path computation heuristics. In particular, we argue that having an estimate of the blocking probability in each domain can be helpful in determining the path computation effort needed to find an end-to-end path.

1. Introduction

Distributed control plane technologies such as Multiprotocol Label Switching (MPLS) and its generalized version GMPLS [1], [2] are opening the door to an array of end-to-end QoS-based services that were previously hard to provide over the shared Internet. The global adoption of service delivery using these technologies assumes that users can be provided connections with well-defined attributes that will not change over the service delivery period with changes in network or user population. Fundamental to this assumption is the ability to dynamically compute routes through the network that satisfy administrative, resource or other types of constraints. Constrained routing, or *path computation*, is an essential functionality in MPLS, GMPLS, or any control plane architecture with end-to-end performance objective.

Following the "distributed intelligence" design philosophy of IP networks, the MPLS architecture [1] viewed path computation as a simple extension of the shortest path first (SPF) algorithm. Almost all MPLS implementations today run a *constrained* shortest path first (CSPF) algorithm that simply accepts or rejects the network links used in shortest path computation based on a set of boolean traffic engineering constraints. This early view of 'on-board' path computation has now been challenged in many ways.

To begin with, path computation requests have become more complex since the early days of MPLS. With GMPLS being applicable to a wide variety of switching technologies such as packet, TDM and wavelength switching, path computation constraints are now more complex than simple bandwidth availability or administrative constraints in packet-switched networks. For example, path computation in all-optical networks with end-to-end transparency may

This work was supported by National Science Foundation grants 0335230, 0335300 and 0335266.

involve optical characteristics of the fiber links. Another example is computing diverse paths for end-to-end protected services, where diversity must be satisfied with respect to all underlying switching layers (wavelength, fiber, duct…). This is a complex problem for which even heuristic algorithms are computationally intensive. Supporting these complex computation problems at every network node requires a large amount of computing power at *all* nodes.

Another factor challenging the "on-board" path computation paradigm is today's more mature and abstract view towards traffic engineering, which makes it less dependent on link-state routing. While link-state routing remains an important vehicle to share topology and resource information between network nodes, it should not be thought of as the *only way* to disseminate this information, and attention should be also given to a centralized view of traffic engineering where not all network nodes support routing, or not all network nodes are able to or decide to access or disseminate traffic engineering information through a link-state protocol. In such scenarios, even the basic topology and resource information needed for path computation can be unavailable to the node initiating a connection.

Finally, the routing architecture of GMPLS is now challenged by the growing interest in GMPLS deployment across multiple *domains*. A domain is any collection of network elements within a common sphere of address management. Examples of domains include an OSPF area, an Autonomous System (AS) running a single instance of OSPF, and multiple autonomous systems within a single service provider's network [3]. Establishing end-to-end connections across a large number of domains would be impractical if not impossible, without establishing a hierarchy across domains. As shown in Fig. 1, by separating the control plane into *path computation* and *signaling planes* these two functionalities can have different architectures with respect to hierarchy. Specifically, the path computation plane can have a hierarchical structure, allowing aggregation of traffic engineering information in each domain into compact models suitable for path computation (with aggregation applied at each level of hierarchy), and the signaling plane can retain its flat architecture and follow the (maybe loosely) computed path in a sequential manner. Hierarchical path computation is also consistent with the routing architecture and requirements set forth by ITU-T for Automatically Switched Optical Networks (ASON) [4], [5].

Because of the above reasons and other motivations listed in [3], there is a growing interest in a control plane element that is dedicated to path computation and can reside in or outside of a network element[1].

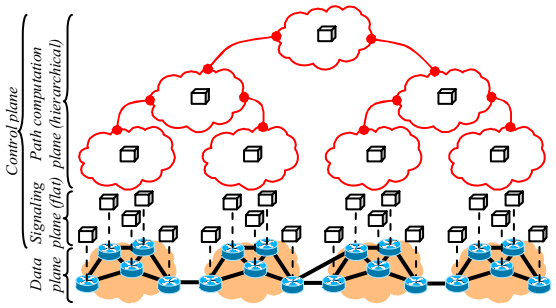

Figure 1. Decomposition of the control plane into a flat signaling plane and a hierarchical path computation plane.

Among the many applications and drivers for dedicated path computation elements, *inter-domain path computation*, i.e., finding end-to-end paths across multiple domains through collaboration between path computation elements, is probably the most important. In this paper, we are particularly interested in *cooperative* inter-domain path computation, where path computation elements in different domains participate in computing an end-to-end path, in full detail or loosely defined, often *prior to signaling the path*. The distinction about *when* PCEs in different domains collaborate is important, as every inter-domain path computation indeed requires one form of collaboration among PCEs; however, not all these forms are considered to be cooperation, as we describe in the next section. Finally, our study is focused on computing a *single* path across multiple domains; therefore, problems such as computing disjoint paths across multiple domains are not considered here.

The outline of the paper is as follows: In Section 2, we give an overview of cooperative path computation. Specifically, we show that cooperation can take one of two forms, which we refer to as *model-based* and *ad hoc*. Through an example we show how these two cooperation modes are different, and also the trade-offs for each mode. We study the model-based cooperation in Section 3. Here, we demonstrate through a probabilistic analysis that by distributing information about congestion, or more precisely blocking in each domain, end-to-end path computation can be made more efficient in terms of the overall path computation *effort*. We briefly discuss ad hoc cooperation in Section 4, just enough to demonstrate that the sizeable literature on ad hoc routing in wireless networks is also

[1] At the time of this writing, the IETF *Path Computation Element* (PCE) working group has finished defining the architectural requirements of a dedicated path computation element [3] and is developing a protocol to interface with the element.

applicable to inter-domain path computation, and probably with a better performance, given the perfect medium (compared to wireless networks) available to PCEs to communicate with each other. Section 5 concludes the paper with our final thoughts.

2. Cooperative path computation

Cooperative path computation is one of several possible ways that path computation elements in different domains can collaborate to compute an end-to-end path. Consider an end-to-end path computation problem or *instance*, and assume that the detailed path information from the source node to the *ingress* of a certain domain (the "current domain") is known[2]. We say a PCE in the current domain and a group of PCEs in other domains have *cooperated* when they exchange path computation information that would *determine the path beyond the ingress to the next domain*. For example, PCEs may cooperate to find the ingress to the next *two domains*, and once the path has reached the second domain ingress, they may cooperate again to find the subsequent domains. They may cooperate to find *all domains* along the path to destination, with full path information in each domain, or partial path information that will have to be *expanded* in each domain. What is common to all these scenarios is that a group of PCEs exchange information in response to a specific path computation instance, and generate path information that goes beyond the next domain ingress.

Cooperation and its different forms can best be described through an example. Fig. 2 shows a scenario where a path computation problem has been resolved up to the ingress node A_1 in domain A (perhaps a partial path has also been set up from the source node to A_1), and the PCE in domain A (PCE-A) receives a request from A_1 to further add to the path. In response,

- PCE-A may compute the best path through domain A, which implicitly identifies the next domain and its ingress (B_1 for example). There is no cooperation in this case, as PCE-A and other PCEs do not exchange any path information; in fact, the PCEs do not even collaborate in this case.

[2] Domain *ingress* here refers to a network resource belonging to the domain that once determined, would make path computation through the domain independent of the path taken to reach the domain. For example, for a packet-switched domain the ingress is the *node* through which a path enters the domain, but for a wavelength-switched domain with optical transparency the ingress is the *wavelength channel* through which the path enters the domain, because one also needs to know the wavelength to be able to continue the path computation through the domain.

- PCE-A may compute the best path through domain A, as well as the best path through the next domain (domain B), or even other domains all the way to the destination. The path information outside domain A may be as detailed as the complete list of all nodes and links, or may be less specific and include only the list of domains and their ingress and egress points. Since computing the path across other domains requires information from other domains, this is clearly an example of collaboration between PCEs; however, depending on how the path is computed it may define cooperation or not, as we describe next.

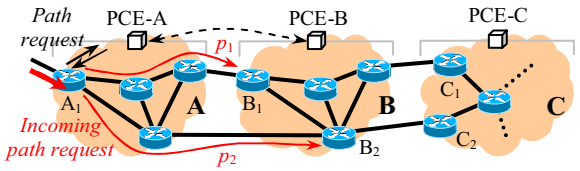

Figure 2. Model-based and ad hoc cooperation examples.

In the second case above, PCE-A may generate the path information in one of several ways:

- It may generate a path based on topology and resource availability information disseminated by other PCEs *before receiving the path computation request*, through periodic or event-based exchanges between domains for example (push model). The level of detail for the generated path depends on the information available to PCE-A, and can range from a simple list of domains, to the full list of links leading to the destination. This is an example of collaboration but no cooperation: Although PCEs are exchanging information, the exchange is not in the context of a path computation instance, and the returned path is unilaterally decided by PCE-A.
- It may compute a loosely defined path consisting of at least the next two domains (using a model of other domains made available as defined in the previous case), and then ask the PCEs in selected domains to return one or more expanded paths across their domains. For example, in Fig. 2, PCE-A may first decide that domain B and domain C would be the next two domains on the way to the destination, using models of these domains available beforehand. Then, in order to select the best path through domain B, PCE-A asks the PCE in domain B (PCE-B) to return a set of acceptable paths (subject to the constraints specified for the end-to-end path) through domain B (e.g., shortest paths from B_1 and B_2 to C_1 and C_2). Once these intra-domain paths are collected, PCE-A possibly

constructs a more detailed graph based on the returned paths, and computes and returns a detailed path going through domain A and B at least. This case is an example of *model-based cooperation*: The set of collaborating PCEs is first determined based on a *model* of other domains, and then PCEs exchange information to solve a specific path computation instance.

- Finally, PCE-A may not have *any* information on other domains, or may decide not to use any information because it is out of date for example. In this case, PCE-A can ask *all* PCEs in reachable neighboring domains to work in parallel and return one or more paths to the destination (again subject to the constraints specified for the end-to-end path). For example, in Fig. 2, recognizing that the destination is not in domain A, PCE-A first tries to compute paths from A_1 to every ingress to neighboring domains (nodes B_1 and B_2 in this case); assuming that both B_1 and B_2 are reachable (via paths p_1 and p_2 respectively), PCE-A then asks PCE-B for paths to the destination starting from B_1 and B_2. PCE-B repeats the process, and requests its own neighboring domains to return the best path to the destination. If PCE-B receives a path to destination, it will send the path back to PCE-A. PCE-A can then choose the best path to the destination among possibly multiple paths that it receives. This case is an example of *ad-hoc cooperation*: Unlike the previous case, the group of collaborating PCEs is not determined *a priori*, but the PCEs still exchange information to solve a specific path computation instance.

The choice between model-based or ad hoc cooperation is determined by several factors, including the number of domains, and more importantly whether domains exchange topology and resource information or not. Model-based cooperation clearly scales better, and perhaps is the only choice when the number of domains is large. Its performance largely depends on the topology and resource information available to each domain, more precisely, how detailed these models are, and how fast they are updated. Ad hoc cooperation on the other hand is best suited for smaller number of domains with no inter-domain topology and resource information exchange.

It should be clear from the above example that cooperative path computation works best when PCEs in different domains can reach each other faster than the speed the network nodes can reach each other through signaling. As discussed, PCEs in different domains may form their own private network to exchange path information, try different paths, and cooperatively decide on an end-to-end path, all without unnecessary signaling attempts that would otherwise perturb the data plane and cause unnecessary updates to routing and traffic engineering information in each domain. Finally, in model-based cooperation PCEs must be able to exchange topology and resource information, most likely in an aggregate form, and update the information over time. One solution is to run a separate instance of a link-state protocol such as OSPF between PCEs in different domains and exchange domain information through opaque link-state advertisements (LSAs) [8], [10].

3. Model-based cooperation

In model-based cooperation, the domains and the order in which they are traversed are determined first, or known in advance through policies or configuration. As a result, the end-to-end path computation becomes a *multistage decision problem*, where each decision stage corresponds to selecting one of the possibly several *inter-domain alternatives*. Depending on what domains represent with regards to the switching technology (i.e., packet, time slot, wavelength, etc.) and routing scope (e.g., an OSPF area or an Autonomous System), inter-domain alternatives represent network nodes, links, or link resources such as time slots and wavelength channels connecting the domains. For example, for packet-switched domains representing OSPF areas, each inter-domain alternative corresponds to an area border router (ABR). For packet-switched domains representing autonomous systems, each inter-domain alternative corresponds to a *pair* of autonomous system border routers (ASBRs) sitting at the endpoints of a link connecting the autonomous systems. For wavelength-switched domains, each inter-domain alternative is a *wavelength channel* connecting the two optical switches at the inter-domain link endpoints. Fig. 3 illustrates these examples.

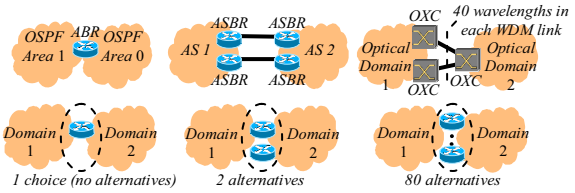

Figure 3. Examples of inter-domain alternatives.

Consider the scenario shown in Fig. 4, where the traversed domains are numbered from 0 to $m \geq 1$. The source node S is located in domain 0 and the destination node D is located in domain m. Assume there are w_i inter-domain alternatives for leaving domain i-1 to domain i. Further assume that a path

computation attempt through domain i fails with the probability of α_i, i.e., given an arbitrary ingress and an arbitrary egress in domain i, there is a chance of α_i that no path will be available between them.

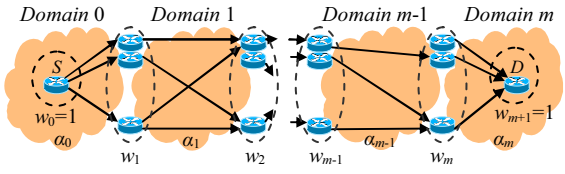

Figure 4. Path computation as a multistage decision problem.

Define $p(i,n)$ i=1,…,m, n=1,…,w_i as the probability of reaching n inter-domain alternatives at stage i, i.e., after domain i-1 and before domain i. By definition, $p(0,n)$=1 for n=1 and $p(0,n)$=0 otherwise. Assuming there are k inter-domain alternatives at stage i-1, the conditional probability of reaching n alternatives at stage i has a binomial distribution with failure probability of $\alpha_{i-1}^{\ k}$. Thus, the unconditional probability $p(i,n)$ can be expanded as

$$p(i,n)=\sum_{k=0}^{w_{i-1}} C_{w_{i-1}}^{n}\, p(i-1,k)\alpha_{i-1}^{(w_{i-1}-n)k}(1-\alpha_{i-1}^{k})^{n} \qquad (1)$$

for i=1,…m and n=1,..,w_i. Using (1), one can compute the distribution of the number of reachable inter-domain alternatives at every stage, starting from the first stage. In particular, the distribution at stage m can be used to compute the probability of failure p_f (or probability of success p_s=1-p_f) for the end-to-end path computation, as follows:

$$p_f = p(m+1,0)=\sum_{k=0}^{w_m} p(m,k)\alpha_m^k \qquad (2\text{-a})$$

$$p_s = 1-p_f = p(m+1,1)=\sum_{k=0}^{w_m} p(m,k)(1-\alpha_m^k) \qquad (2\text{-b})$$

The distribution of reachable alternatives is binomial at the first stage, but gets more involved after each stage[3]. To gain better understanding of the dynamics of this distribution, i.e., how the distribution evolves from one inter-domain stage to the next, let us analyze a simple case where the number of inter-domain alternatives (reachable and unreachable) at every stage is the same, i.e., w_1=w_2=…=w_m=w. Let the (w+1)×1 vector $\mathbf{p}_k$=($p(k,0)$, $p(k,1)$,…, $p(k,w)$)^T denote the distribution of the number of reachable alternatives at stage k. It follows from (1) that $\mathbf{p}_k$ is the state vector of an unforced linear dynamical system with the state equation and initial condition given by

$$\mathbf{p}_k = \mathbf{A}_{k-1}\mathbf{p}_{k-1} \quad ;k=1,\ldots,m \qquad (3\text{-a})$$

$$\mathbf{p}_0 = (0\ 1\ 0\ \cdots\ 0)^T \qquad (3\text{-b})$$

where the elements of the (w+1)×(w+1) state matrices $\mathbf{A}_k$=[a_{ijk}] k=0,…,m-1 are given by

$$a_{ijk} = C_w^i \alpha_k^{(w-i)j}(1-\alpha_k^j)^i \;; i=0,...,w;\ j=0,...,w \qquad (3\text{-c})$$

with $a_{0,0,k}$=1. We call the linear system in (3) the *stage distribution system*. Now consider a scalar output of the stage distribution system defined as

$$y_k = \mathbf{c}_k \cdot \mathbf{p}_k \quad ;k=0,1,\ldots,m \qquad (4\text{-a})$$

$$\mathbf{c}_k = (1\ \alpha_k\ \cdots\ \alpha_k^W) \quad ;k=0,1,...,m \qquad (4\text{-b})$$

It follows from (2) that the probability of failure for the end-to-end path computation is given by the system output at sample (stage) m, i.e., p_f=y_m. For this reason, we call the system output at sample k the *projected probability of failure* at sample (stage) k.

As one can expect, the evolution of the distribution if inter-domain alternatives and the probability of failure for the end-to-end path computation as the number of domains grows is intimately related to the properties of the state matrices $\mathbf{A}_k$, k=0,…,m-1 and the output vectors $\mathbf{c}_k$, k=0,…,m. Two immediate observations follow:

- The eigenvalues of all $\mathbf{A}_k$ matrices are inside or on the unit circle in the complex plane, and therefore the stage distribution system is *marginally stable*. This is an immediate result of the Gershgorin's Circle Theorem [6] and the fact that all columns in $\mathbf{A}_k$ matrices have a sum equal to one. This is rather trivial, as the state vector represents a probability distribution, which by definition is bounded. Furthermore, all $\mathbf{A}_k$ matrices have *at least* one eigenvalue equal to one, corresponding to the "trivial" eigenvector $\mathbf{p}_{\text{null}}$=(1 0 0 … 0)^T. This follows from the fact that if no alternatives are reachable at stage k-1, then no alternatives are reachable at stage k, i.e., $\mathbf{A}_k.\mathbf{p}_{\text{null}}$=$\mathbf{p}_{\text{null}}$.
- A less trivial observation is that *the stage distribution system with the observed output (4) is observable* [7], meaning that one can compute the exact distribution of reachable alternatives at stages 0 through k≤m, *using only the projected probability of failure at these stages*. The full implications of this observation need to be studied in more detail, but one immediate consequence is that the domains need to exchange only the projected probability of failure and the blocking probability in each domain instead of the entire distribution vector.

[3] However, the *conditional* distribution, i.e., the distribution of the number of alternatives at stage i given the number of alternatives at stage i-1, remains binomial.

Fig. 5 shows an example of the stage distribution system dynamics. In this example, there are m=100 inter-domain stages (101 domains), all domains have the same blocking probability (and therefore the same state matrix $\mathbf{A}_k$=$\mathbf{A}$, k=0,…,m-1), and there are w=4 alternatives at every stage. The state vector $\mathbf{p}_k$ has w+1=5 states, which are the probabilities of reaching 0, 1, 2, 3 or 4 inter-domain alternatives at stage k. Each chart in Fig. 5 shows the evolution of $\mathbf{p}_k$ elements as path computation advances from one domain to another. For example, in Fig. 5(a), which corresponds to the blocking probability of α=0.25 in each domain, we see it is most likely to reach n=3 alternatives at the first stage (note the initial peak of about 0.42 at n=3), but after the second or third stage, *all* alternatives at each stage can most likely be reached (the peak of the distribution shifts from n=3 to n=4). Increasing the blocking probability in each domain to α=0.40, we see in Fig. 5(b) that the most likely number of alternatives to be reached after a few stages is still 4, although the probability of reaching 4 alternatives is smaller than the previous case.

It may seem from Fig. 5(a) and Fig. 5(b) that the stage distribution system reaches a steady-state. However, numerical computation of the distribution for higher number of domains (1000 and above) shows that the probability of reaching all alternatives ultimately falls to zero, and the probability of reaching no alternative ultimately reaches one, although very slowly. This slow convergence can be explained by the fact that the state matrix **A** in these two cases has a *second* eigenvalue very close to 1 (eigenvalues for each case are also shown in the figure).

The convergence is much faster at higher blocking probabilities, as shown in Fig. 5(c) and Fig. 5(d). Here, the probability of reaching any number of alternatives n>0 quickly gets smaller at each stage. This is also evident from Fig. 6, where we have plotted the projected probability of failure at each inter-domain stage: In the first two cases (α=0.25 and α=0.40) the projected probability of failure *almost* converges to a steady-state probability strictly smaller than one after a few stages (although we repeat that should the number of domains go up all projected probabilities ultimately approach one), but in the next two cases (α=0.65 and α=0.80) the projected probability of failure has a quick convergence to one.

Knowing the blocking probability of other domains, a path computation element can benefit from this type of analysis in several ways. For example, before attempting cooperation, a PCE can estimate the probability of failure for end-to-end path computation, and reject the request without starting any cooperation.

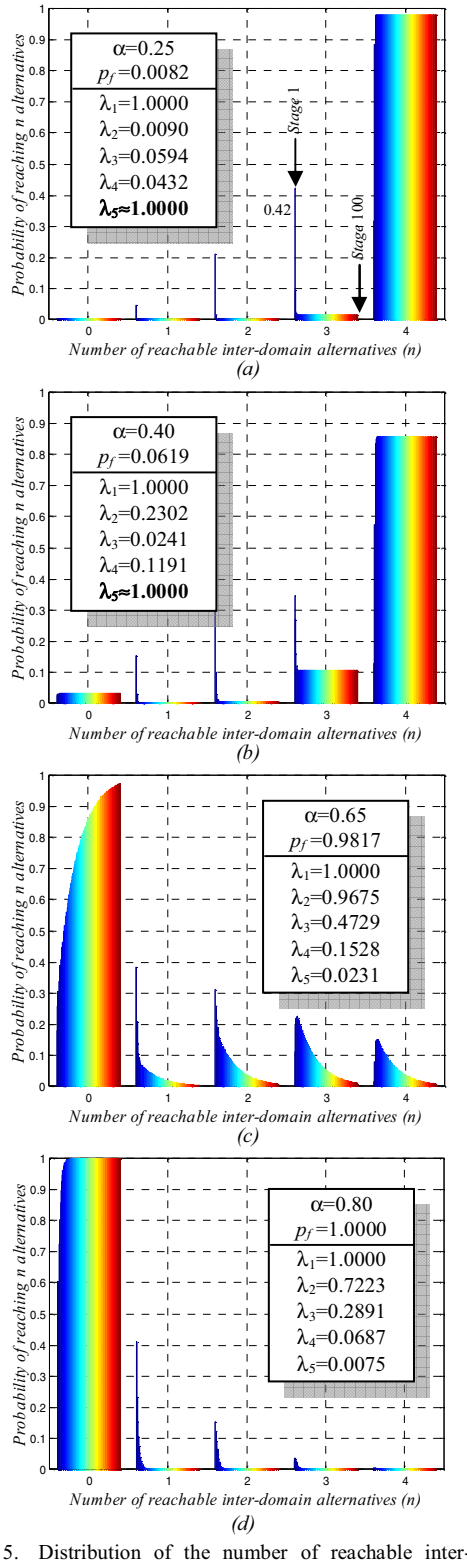

Figure 5. Distribution of the number of reachable inter-domain alternatives at each stage for a total of m=100 stages and w=4 inetr-domain alternatives; each chart corresponds to a different blocking probibility (blocking probabilities are assumed to be the same for all domains).

A more interesting application is *optimizing the path computation effort*: Instead of choosing all inter-domain alternatives (which can be a few hundred in case of wavelengths in optical domains for example), a PCE can start the cooperation using a certain number of alternatives that would result in a given probability of success for end-to-end path computation. For example, knowing that the end-to-end path computation with 4 inter-domain alternatives at each stage succeeds with a probability of 94% or better ($p_f \leq 0.0619$) when blocking probability in each domain is no more than 0.4, a cooperating PCE can limit the number of considered alternatives to 4, even if there are more alternatives.

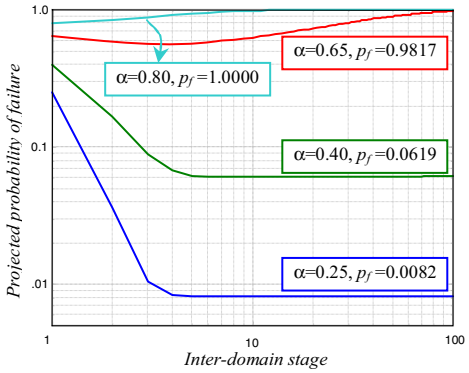

Figure 6. Projected probability of failure.

4. Ad hoc cooperation

Ad hoc cooperation requires no exchange of topology and resource information between domains. Therefore, it is particularly effective for a small number of domains with PCEs that have limited visibility of each other and are not exchanging any topology or resource information (this could be because of lack of support for PCE-PCE communication, or because of trust or policy issues for example). Since the same assumptions hold true for wireless networks, ad hoc routing protocols developed in the context of wireless networks could also be applicable to inter-domain path computation. In fact, such protocols should have even a better performance here, as communication between PCEs is more reliable than communication through wireless channels.

One example of an ad hoc routing protocol is *Dynamic Source Routing (DSR)* [11], [12]. In dynamic source routing the source node broadcasts a *route request* (RREQ) message in search of a path towards a specific destination. Each node receiving a route request discards the request if it has already been processed, broadcasts the request again (each node only once) if a route to destination is unavailable, or returns the route in a *route reply* (RREP) message if it has a route to the destination node.

Fig. 7 shows an example of ad hoc cooperation using an algorithm similar to DSR. In this example, path computation elements reside in border network elements (nodes A_1, A_2, B_1, B_2, B_3, C_1, C_2, D_1 and D_2 in the figure) and have limited visibility of each other; in particular, each PCE can only communicate with its *peer* PCE in the neighboring domain, or other PCEs in the same domain. In an effort to compute a path to destination node D, the source node S sends separate *path request* messages to both A_1 and A_2 PCEs. A path request message sent to any node *N* includes the destination, the traffic engineering constraints for the end-to-end path, the best path to reach *N* from the source node, and the cost of this path. A_1 and A_2 PCEs compute the best path from S if not provided already, and send their own path request to their peer PCEs, B_1 and C_1 in this example. Upon receiving a path request message, each PCE

- Computes the best path to destination if it is in the same domain as the destination, and returns the end-to-end path to the PCE which sent the path request, or
- If receiving the path request from a peer PCE in a *different* domain which shows a lower cost of reaching the PCE from source, computes a path to every domain egress subject to given constraints, and if successful updates the path request message, and forwards the message to the domain egress, or
- If receiving the path request from a PCE in the same domain which shows a lower cost of reaching the PCE from source, computes a path to every peer PCE in other domains subject to given constraints, and if successful updates the path request message and forwards the message to the peer PCE.

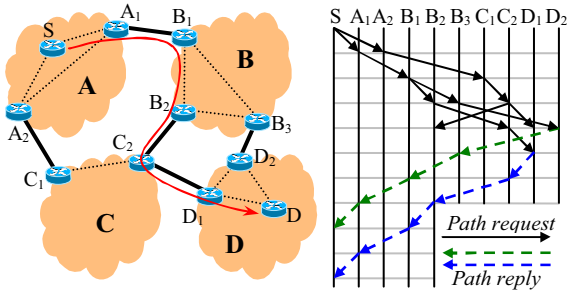

Figure 7. Example of ad hoc cooperative path computation.

As shown in Fig. 7, the source node may receive multiple paths to the destination. In this case the source

node may decide to wait to receive multiple paths and pick the best one, or may decide to signal the first path it receives to reduce the path set up time. The trade-off between path computation latency and the quality of paths that are signaled needs to be studied in detail.

5. Conclusion

Cooperation is a term that has been loosely used to refer to collaboration between path computation elements in the context of a single path computation instance. We offered a more accurate description of cooperation, and identified two forms of cooperation, called model-based and ad hoc. These models are not mutually exclusive, and an end-to-end path computation can indeed use both models to complete the path. Model-based cooperation is an example of divide-and-conquer algorithm: It first places a certain order on the traversed domains through a high-level path computation using full or aggregate models of other domains, and then finds the best path through the selected domains. When aggregate domain models are used, because of scalability or trust and privacy reasons for example, we showed it is beneficial to also advertise a measure of blocking in each domain to guide the path computation elements in choosing the appropriate path computation effort in terms of *number of inter-domain alternatives* that they would have to consider. Ad hoc cooperation is a less explored alternative to cooperation. While ad hoc routing protocols developed for wireless networks can provide good insight into how ad hoc cooperation should work, including traffic engineering information in the routing process will have an impact on the existing ad hoc protocols that needs to be studied in more detail.

6. Acknowledgement

We acknowledge Sameer Sharma for his work on model-based path computation examples.

7. References

[1] E. Rosen, A. Viswanathan and R. Callon, "Multiprotocol Label Switching Architecture," IETF RFC 3031.

[2] E. Mannie et al., "Generalized Multi-Protocol Label Switching (GMPLS) architecture," IETF RFC 3945.

[3] Farrel, J. P. Vasseur and J. Ash, "A Path Computation Element (PCE) based architecture," Internet draft draft-ietf-pce-architecture.

[4] "Architecture and requirements for routing in the automatically switched optical networks," ITU-T recommendation G.7715, July 2002.

[5] "Architecture for the automatically switched optical network (ASON)," ITU-T recommendation G.8080, November 2001.

[6] G. H. Golub and C. F. Van Loan, Matrix Computations, Third Edition. Baltimore, MD: Johns Hopkins University Press, 1996.

[7] P. Sage and C. C. White III, Optimum Systems Control, Second Edition. Englewood Cliffs: NJ: Prentice-Hall, Inc., 1977.

[8] S. Sharma and B. Jabbari, "NARB: Network Access Resource Broker," George Mason's Communications and Networking Lab (CNL) technical report, 2003.

[9] Xi Yang et al., "Policy-based resource management and service provisioning in GMPLS networks," First IEEE Workshop on Adaptive Policy-based Management in Network Management and Control (A-PBM), Barcelona, Spain, April 2006.

[10] T. Lehman, J. Sobieski and B. Jabbari, "DRAGON: A framework for service provisioning in heterogeneous grid networks," IEEE Communications Magazine, pp. 84-90, March 2006.

[11] E. Perkins, Editor, Ad Hoc Networking. Boston, MA; Addison-Wesley, 2001.

[12] David B. Johnson, David A. Maltz and Yih-Chun Hu, "The Dynamic Source Routing protocol (DSR) for mobile ad hoc networks," Internet draft draft-ietf-manet-dsr.