A distributed algorithm for inter-domain resources provisioning

Marc-Antoine Weisser and Joanna Tomasik
Computer Science Department, Supélec
3, rue Joliot-Curie, 91192 Gif-sur-Yvette cedex, France
Email: {Marc-Antoine.Weisser, Joanna.Tomasik}@supelec.fr

***Abstract*— The introduction of QoS guarantees from peer to peer in the Internet requires a mechanism for finding paths in the inter-domain network. This mechanism has to be distributed, support classical constraints (delay, bandwidth, jitter, loss rate, administrative cost) and be able to find multi-constraint paths. The second problem is hard even if global information concerning the resources available in the network is known, which is impossible in the Internet. We are convinced that RSVP would not be an efficient solution in this context. We propose an alternative solution better adapted for fulfilling the QoS requirements. It is heuristic and it makes decisions according to routing tables stored in border routers. The obtained simulation results have shown that its performance is very satisfactory.**

I. Introduction

The Internet is a network based on IP (*Internet Protocol*). Routing between independent sub-networks (domains) is assured by BGP (*Border Gateway Protocol*) which uses IP. In the inter-domain network, there was no need for QoS (*Quality of Service*) guarantees when these two protocols were drawn up and these guarantees are not integrated within the Internet. Today, the expectations of the network users have changed. The introduction of new applications, like voice-over-IP, video-conference or e-commerce forces the providers to adapt their networks to include the QoS requirements. It is widely accepted that resource reservation provides strong QoS guarantees from peer to peer and not only probabilistic guarantees such as an access protocol on the borders between the domains might give.

The existing reservation protocol RSVP (*ReSerVation Protocol*) [1] was designed for an IP network considered as a single domain network. RSVP is not scalable to the inter-domain level. The scalability problem occurs because RSVP treats each flow individually. Indeed for each flow, RSVP has to:

- create for each router a *soft state table* containing flow information;
- dedicate a leaky bucket mechanism to control the traffic of each flow for each router;
- send *refresh messages* periodically to keep the reservation alive.

RSVP identifies a flow by its sender and destination addresses. Such an approach forces the protocol to deal with a great number of flows. Notice that single flow reservation demands can significantly vary. In particular, RSVP may reserve paths for a great number of "small" flows. In the inter-domain context, the need for reserving path "flow by flow" makes the use of RSVP unrealistic. The inter-domain resources reservation needs introduction of a flow aggregation mechanism.

Let us take the hypothesis that this type of mechanism is implemented in the inter-domain routers. To establish a reserved path for aggregated flows, we will need a protocol which :

1) does not have to know the global network state, meaning the resources available in all the links and all the routers;
2) finds paths which satisfy a set of criteria like delay, costs, bandwidth, jitter, lost rate;
3) can create reserved paths independently from BGP routing tables.

Points 2 and 3 listed above eliminate RSVP from consideration. Indeed, RSVP uses only the BGP routing tables to construct the reserved path. Moreover, the only constraints integrated today in this protocol are delay and bandwidth.

Furthermore, RSVP uses an upstream reservation paradigm. In our opinion a downstream reservation approach will be more efficient because in the context

This work was partially founded by Research and Development Department of France Telecom.

of the inter-domain routing there is no information concerning the used resources. Before asking a router whether a reservation is possible, there is no way to know if the router will accept or reject the demand.

In the RSVP approach, the algorithm finds a complete path from the source to the receiver and then asks each router on the path if the demand is accepted. The path's finding is downstream, but the reservation process is upstream. If one of the routers rejects the demand, the whole path is rejected and the reservation procedure has to be restarted from the beginning. As we said, RSVP uses the routing table of BGP to find the path. The path is created independently of the constraints, so it may be inadequate to QoS requests. It seems more efficient to construct the path downstream and to reserve the resources simultaneously. Indeed, it is possible to verify hop by hop that the constraints are still satisfied. If they are not, the algorithm can return one hop and try another solution without rejecting the whole path. Finally, RSVP allows the receiver to choose QoS level. In the downstream reservation approach, it could also be possible for the receiver to specify its QoS level. In this case the sender has to ask the receiver about the QoS request at the beginning of the reservation. The drawback is that it increases the number of messages by one exchange although it can improve searching for the path. We think that it is a negligeable loss in comparison with the difficulties of the multi-constraint path finding.

In the next section, we will give more details concerning our goal. In section III, we present the multi-constraint path problem and its modelisation. In section IV, we propose an heuristic algorithm to solve this problem and we study its complexity. Section V contains simulation model and the results. We formulate the conclusions about our algorithm performance in the last section.

II. Goals of the inter-domain QoS routing

Our work focuses on the inter-domain routing including a mechanism of aggregation. Our goal is to provide an algorithm finding multi-constraint paths from a router in a domain to another router in another domain. In this context, the algorithm has to satisfy some requirements. Providers want to limit the broadcast of information about their domain topologies. For this reason the algorithm cannot be centralized or semi-distributed such as the ones proposed for the bandwidth allocation [2], [3], [4] or the multi-constrained path reservation [5]. Our algorithm has to be distributed. In a distributed approach presented in [6], control packets collect measurements in a network. We think that operators do not want to allow external agents to measure their domains' performence parameters. Moreover, this solution does not require a found route to satisfy strictly each of the given constraints.

The network's state, ie. the resources available on links and in routers, is continuously changing. Today, the overhead caused by the number of messages for each routers in order to keep the global state of the network is too important. Therefore algorithms such as [7] are not adapted, because they use the knowledge of the global network's state. Our algorithm does not have to use this knowledge. As domains are managed independently by their operators, it may happen that the proposed reservation protocol is not applied in some of them. The algorithm should be able to work and to reserve a path excluding domains which do not use it.

Finally, new needs for QoS may appear with coming into view of new applications. The algorithm has to be adaptable to take into account emerging constraints. In the context of the networks, the algorithm has to work with contraints which are additive, multiplicative and convex. Well known constraints are: delay (additive), cost (additive), bandwidth (convex), lost rate (multiplicative), jitter (additive).

The problem of finding multi-constraint paths with more than one additive and multiplicative constraint in a network is a *NP*-complete problem [8]. This result implies that, even in a centralized model, to compute a path in a polynomial time, we need to use an heuristic algorithm. The lack of complete network's state information leads us to choose a probabilistic approach.

We will present an algorithm finding multi-constraint paths which satisfy the following properties :

- it is distributed;
- it is able to work without global information about the resources available in the links and in the routers;
- its installation can be incremental, this means that it works even if it is deployed in a subset of the domains;
- it can support any additive, multiplicative or convex constraints;
- it works in polynomial time, ie. in each node the algorithm is in polynomial time and the number of communication steps is also polynomial;
- it is a *Las Vegas* algorithm, this means that it produces correct results but does not always find a solution (even if it exists).

Our algorithm is based on a work presented in [9].

The main idea is to send a *probe message* from a source router within a domain to a destination router in another domain. The probe is going from domain to domain through the network. In each visited domain, the probe demands a reservation. The reserved path is constructed hop by hop. When the probe reaches its destination, a *validating message* comes back to the source validating the reservation demand made by the probe message. The probe has a limit on the number of domains visited. If the number of visited domains becomes greater than this limit, a *failure message* is send back to the source, discarding the reservation demand.

Because our algorithm is not deterministic, we need simulations to validate it. We have developed a simulator to model the behavior of the algorithm.

III. The multi-constraint path problem

We focus on the inter-domain network. We consider a graph $G = (V, A)$. V is a set of vertices, each vertex represents a domain in the Internet. A is a set of arcs which represent connections between domains.

We define K weight functions on the arcs $w_i : A \rightarrow \mathbb{N}$, $i \in [1; K]$. These functions represent the global characteristics of a link (physical proprieties such as the delay of propagation or the available bandwidth) and of a domain (the QoS guarantees which a domain can provide). There are three types of weights :

- an additive weight for delay, cost, and jitter;
- a multiplicative weight for loss rate;
- a convex weight for bandwidth.

Let $W_i : path(G) \rightarrow \mathbb{N}$, $i \in [1; K]$ be a weight function on the paths of G. For an additive characteristic, the weight function of a path is defined as a sum of arc weigths:

$$W_i(p) = \sum_{u \in p} w_i(u)$$

We can transform a multiplicative weight into an additive one using a logarithm. For a convex characteristic, weight functions are:

$$W_i(p) = \min_{u \in p} w_i(u)$$

Finally, we define K satisfaction functions, $S_i : path(G) \times \mathbb{N} \rightarrow \{true, false\}$. For arguments these functions have a path and a constraint. They return true if the path satisfies a given constraint and false otherwise. We use these functions to test if a path satisfies given constraints. For additive and multiplicative constraints; the functions are :

$$S_j(p, c) = \begin{cases} true, & \textbf{if } W_j(p) \leq c \\ false, & \textbf{otherwise} \end{cases}$$

and for convex constraints :

$$S_{j'}(p, c) = \begin{cases} true, & \textbf{if } W_{j'}(p) \geq c \\ false, & \textbf{otherwise} \end{cases}$$

where p is a path and c is the demanded value of the j^{th} constraint.

Prob. 1 The multi-constraint path problem

Entries:

- a graph $G = (V, A)$;
- a set of weight functions on the arcs: w_i;
- a set of weight functions on the paths: W_i;
- a set of satisfaction functions: S_i;
- $v_o, v_d \in V$, the origin and the destination;
- a set of K constraints $c_i \in \mathbb{N}$.

Question: Find a path p from v_o to v_d for which

$$\forall i \in [1; K], S_i(W_i(p), c_i) = \textbf{true}$$

We use the definitions given above to describe the multi-constraint path problem: Prob. 1. In the next section we propose an heuristic algorithm to solve it.

IV. Proposed heuristic solution

Korkmaz and Krunz presented in [9] an heuristic algorithm to find a multi-constraint path in a graph. Their approach is based on a depth first search algorithm with some cut mechanisms. These mechanisms use the shortest path matrix for each constraint. The Internet's protocols do not provide this knowledge. The only quite reliable information available in the global network is the number of domains between a source and a destination. The information needed for the QoS implementation such as delay or bandwidth are unknown. There are some suggestions, for instance, to estimate the available bandwidth [10] but no solution has been commonly accepted.

The design of the algorithm proposed in our article is inspired by the one described in [9]. The advantage of our algorithm is that we do not need to use distant matrices for each constraint.

A. Algorithm description

Our distributed algorithm uses three types of messages. Probe messages are used to find a path from the source to the destination. These paths are found hop by hop and their descriptions are stored into the probes. A current path (between the source and the node by which actually a probe is passing) satisfies the demanded

QoS parameters. For a given request, a visited node is a node by which the probe passed once. The visited nodes are stored in a field of the probe. Two mechanisms are used to control the search time: a node is never visited more than once and a maximum number of visited nodes is set in advance. This parameter is noted N_{hop}. The two other messages types are acknowledgments: the *validating ACK* and the *aborting ACK*. They are used to validate and definitively accept resources allocation or to abort and reject ressources allocation.

Let us describe the algorithm more precisely. When a source node wants to establish a multi-constraint path to a destination node, it creates a probe message containing:

- the invariant part of the request (origin, destination, set of constraints);
- the current path starting from the origin by which the probe has gone through up to a current node (a node containing the probe);
- the weight of the current path for the different constraints;
- the list of the visited nodes.

A probe is sent into the network to find a multi-constraint path and to reserve the needed resources. It is forwarded from node to node until it reaches the destination or aborts the research. We will discuss the aborting mechanism later. When a probe comes into a non-visited node, the node determines the QoS guarantees (for each constraint) which can be fulfilled for this reservation request. The path and the weights path fields of the probe are updated with these guarantees. If the guarantees do not satisfy all the constraints, then the probe is sent back to the last visited node and the previous values of its fields are restored. Otherwise it is sent randomly to a non-visited node in the neighborhood of the current node. This behavior allows each node to determine if it accepts or not an incoming reservation. Moreover, the QoS parameters have not to be broadcast in the network before the request arrives.

The two following pseudocodes describe the behavior of a node which wants to request a multi-constraint path (Pseudocode 1) and the behavior of the nodes which receive a probe (Pseudocode 2).

A validating ACK is forwarded from the destination node to the source. To go back to the source, it passes by each node included in the probe's path. In each node the demanded resources are definitively reserved. The aborting ACKs also go back to the source but in each node the resource allocations are discarded.

The mechanism which computes the guarantees available in a node is left open. Every node can choose an appropriate mechanism depending on the internal configuration.

Pseudocode 1 Sender protocol

1) Create a probe containing:
   - source v_s;
   - destination v_d;
   - path between the origin and the current position: (v_s);
   - weights for the K constraints of this path: $\{c_1, \ldots, c_K\}$;
   - set of visited nodes: $\{v_s\}$;
   - set of K constraints to satisfy $\{w_1, ...w_K\}$;
   - maximum number of visited nodes allowed: N_{hop}.
2) Choose randomly a node v in the neighborhood according to distribution $\mathcal{D}$.
3) Send the probe to v.
4) **end of the protocol**

Pseudocode 2 Receiver protocol

Current node receiving the probe is noted as v_c and the previous node which sent the probe is noted as v_p.

1) **if** the number of visited nodes is greater than the allowed maximum N_{hop}
   then
   - destroy the probe and send an aborting ACK to v_p.
   - **end of the protocol**
2) Compute and allocate the QoS guarantees available in the node.
3) Compute and upgrade the path and its weights contained in the probe field using the QoS guarantees computed at step 1).
4) **if** all nodes of the neighborhood have been visited **or** if the path's weights do not satisfy the constraints
   then
   - remove v_c from the probe's path and downgrade the path's weights.
   - send the probe back to v_p.
   - **end of the protocol**
5) **if** v_c is the destination
   then
   - destroy the probe and send a validating ACK to v_p.
   - **end of the protocol**
6) Choose randomly, according to a distribution $\mathcal{D}$, a non-visited node v in the neighborhood of v_c.
7) Send the probe to v.
8) **end of the protocol**

In point 4) of Pseudocode 1, we have to delete the last node from the path contained in the probe and to downgrade the path's weights. This operation can be done because the constraints are additive or convex.

B. Number of message exchanges

Since the proposed algorithm is distributed, we have to consider the complexity in terms of the number of exchanged messages. For a given request, the maximum number of exchanged probes is $\mathcal{O}(\min(N_{hop}, |V|))$ because the probe cannot visit a node more than once and

cannot visit more nodes than N_{hop}. The ACK messages are sent back from the destination to the origin. In the worst case, the number of nodes in the current path is equal to N_{hop}. The order of the complexity in number of exchanged messages for one request is given by:

$$\mathcal{O}(\min(N_{hop}, |V|))$$

As usual, a problem to find the mean complexity in number of exchanged messages is a more difficult one. In the case of a simple topology like chain, binary tree or ring, the mean complexity can be computed. In these three cases, the order of the number of exchange messages is linear. To illustrate the complexity of computing the number of exchanges messages on the most general case, we consider a 2-dimension grid and where the constraints are relaxed. In this case all the paths between the origin and the destination are a solution. We consider only the mean number of exchanged messages when a solution has been found. To compute this mean, we have to know the number of the simple paths between two vertices. This problem has already been studied. Liśkiewicz, Ogihara and Toda gave a proof that this problem is #*P*-complete [11].

C. Extensions

We can consider the algorithm described above as a starting point for extensions and modifications. In the first version of the algorithm, the probe cannot be transmited towards a node already visited. This is a strong requirement which reduces the time of the search but which may also reduce the number of accepted requests. A flexible alternative algorithm is to accept the probes to be forwarded to already visited nodes but to outlaw the probes to be forwarded through already visited arcs. In this case the complexity is:

$$\mathcal{O}(\min(N_{hop}, |V \times V|)).$$

In our base algorithm, the probe is sent to any node in the neighborhood. We may prefer to forward the probe only to a node which is closer (in number of hops) to the destination. BGP is based on the shortest hop path. We can use it to determine the preferred nodes even if the "political" and economical relations between domains modify its shortest paths. In this case, we use a shortest distance matrix of the number of hop but we still do not use distance matrices of each constraint.

In our simulations, we will test four versions of our algorithm. The basic one is called `vertex` algorithm. The second one, `arc` algorithm, allows the probe to pass more than once by the same node but not to use more than once the same link. The third and fourth ones, `vertex+` algorithm and `arc+` algorithm, are constructed upon the first and the second ones respectively, but they require that the probes are forwarded to a node closer to the destination.

V. Simulations

A. Plan

To test the algorithm, we run simulations on random topologies representative for inter-domain networks. We use the generator BRITE [12] to create these random topologies. We choose this generator because it is based on the Waxman graphs which are commonly used to create random topologies representing telecommunication networks. Moreover, BRITE is a frequently used tool, so it can be considered as a benchmark. The topologies to be analyzed have 20 and 50 nodes.

BRITE creates topologies with delay and bandwidth parameters attached to the arcs. Delays generated by BRITE depend only on the distance between two nodes (propagation time of a physical medium). For our simulations, we have to introduce a packet delay caused by its passage through a domain. We assume that the delay follows a normal distribution with mean $\mu = 80$ ms and variance $\sigma^2 = 20$ ms. A bandwidth on the arc (distributed uniformly between 1 and 100 MB) represents the maximum bandwidth accepted by a domain for a request. We assume that a domain does not accept the requests with bandwidth smaller than 1 MB because the number of requests to manage will be too important. A domain rejects also demands with bandwidth greater than 100 MB which are too significant.

We started to study the behavior of the reservation algorithm for requests upon one constraint only, the delay. The requests are generated with a demand of delay uniformly distributed on $[x; x + 20]$ ms. We generate $N = 10,000$ requests for the values of x in $\{0; 10; ...; 340\}$. We choose these values because the telephony over IP have to guarantee a delay less than 300 ms. We attempt to satisfy each request applying all the algorithms presented above. In the Internet, the protocol RSVP tries to establish a path using the BGP routing tables. For the four algorithms which we designed, we set N_{hop} to the number of vertices of the graph. Our topologies were generated by BRITE and they do not contain all information needed to implement a full BGP protocol such as economical relationships. Therefore we use a shortest hop path algorithm to simulate RSVP. This algorithm serves us as a benchmark. We have five algorithms to test:

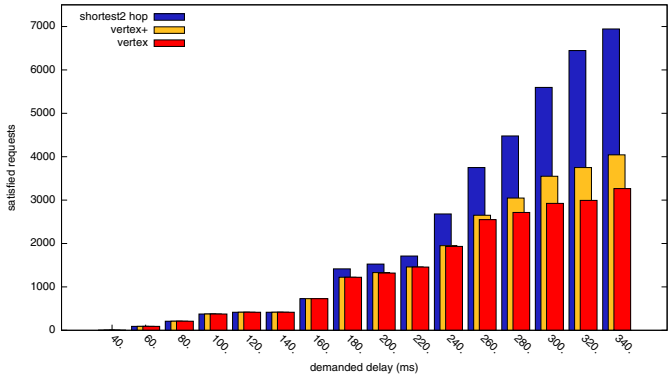

Fig. 1. Number of one constraint (delay) paths found in a 50 nodes Waxman graph in function of demanded delay. Comparaison between `vertex`, `vertex+` and `shortest hop` algorithms.

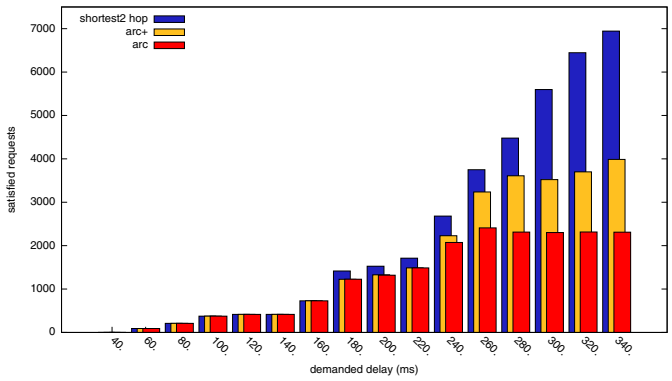

Fig. 2. Number of one constraint (delay) paths found in a 50 nodes Waxman graph in function of demanded delay. Comparaison between `arc`, `arc+` and `shortest hop` algorithms.

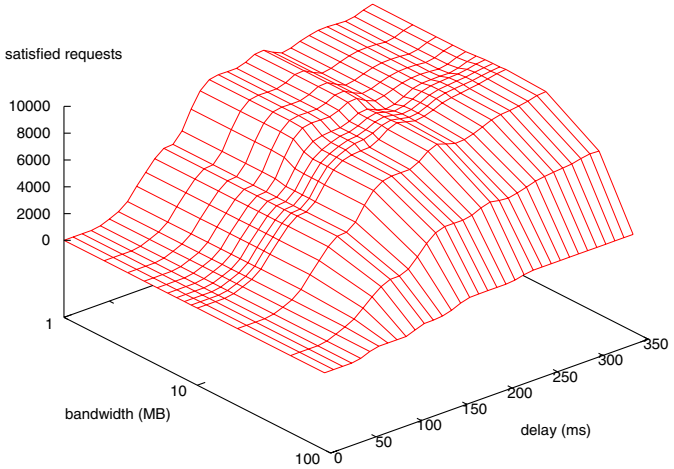

Fig. 3. Number of two constraints (delay and bandwidth) paths found in a 20 nodes Waxman graph.

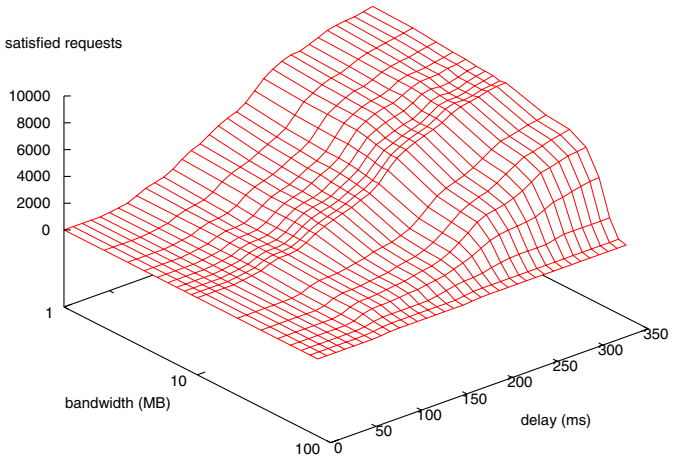

Fig. 4. Number of two constraints (delay and bandwidth) paths found in a 50 nodes Waxman graph.

- `vertex` algorithm;
- `vertex+` algorithm;
- `arc` algorithm;
- `arc+` algorithm;
- `shortest hop` algorithm (RSVP).

As our algorithm is proposed in order to trace multi-constraint paths, we analyze its performance for two crucial QoS parameters: delay and bandwidth. We generate requests with delay and bandwidth uniformly distributed on $[x; x+20]$ ms and $[y; (y+1)]$ MB with $x \in \{0; ...340\}$ and $y \in \{0, 8\}$. We generate $N = 10,000$ requests for each algorithm.

B. One constraint (delay) paths

Fig. 1 and 2 shows the number of paths found for each algorithm depending on the demanded delay. All the algorithms give almost the same results for the strongest constraint requests (with a delay varying from 0 to 230 ms). The best algorithm in this case is the shortest hop algorithm. Its good performance can be explained by the correlation between the path's delay and the nodes number in a path. The worst results are given by the `arc` algorithm. This fact is not astonishing and can be explained. To limit search time, the probes have a given maximum number of visited arcs but they can visit the same node many times (if they use different arcs to access it). This approach can be advantageous to find a better solution but its major drawback is that the probes may lost themselves in the graph without reaching their destination. The `vertex`, `vertex+` and `arc+` algorithms give better results. In contrast with `arc` algorithm, `arc+` probes do not lost themselves in a network and for this reason its results are better. The topology of the network has an influence upon the algorithm performances. The irregularity of the tested topologies causes the irregularities of the obtained curves.

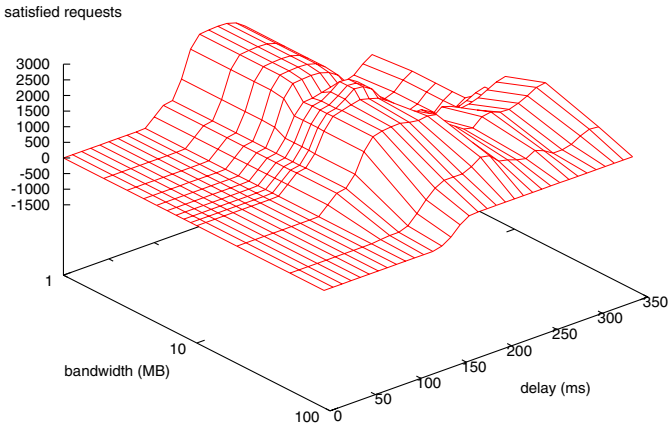

Fig. 5. The difference between the number request accepted by the arc+ and shortest hop algorithm for a graph of size 20.

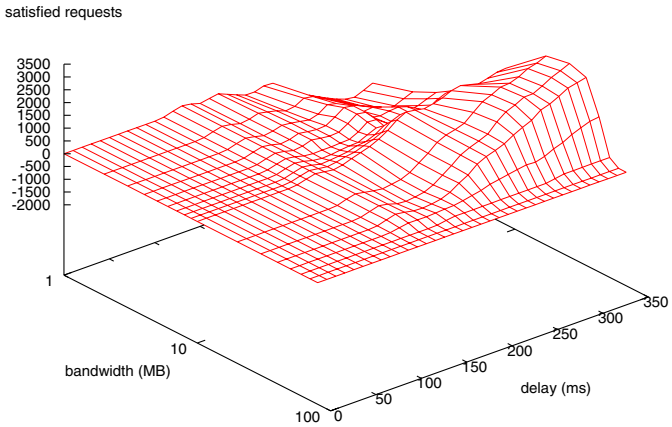

Fig. 7. The difference between the number request accepted by the `arc+` and `shortest hop` algorithms for a graph of size 50.

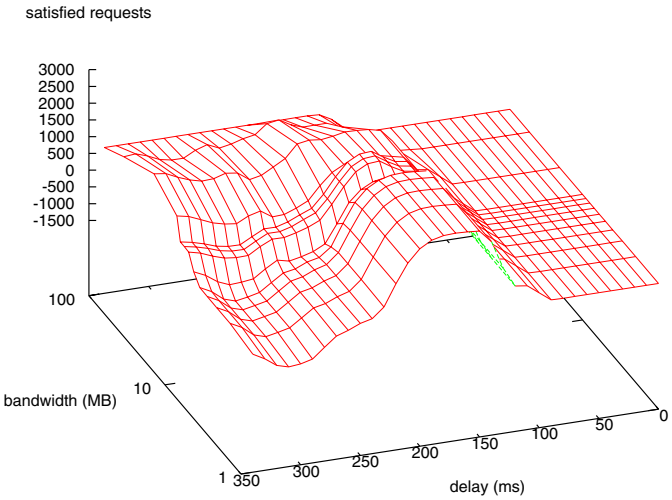

Fig. 6. Back view of the Fig. 5

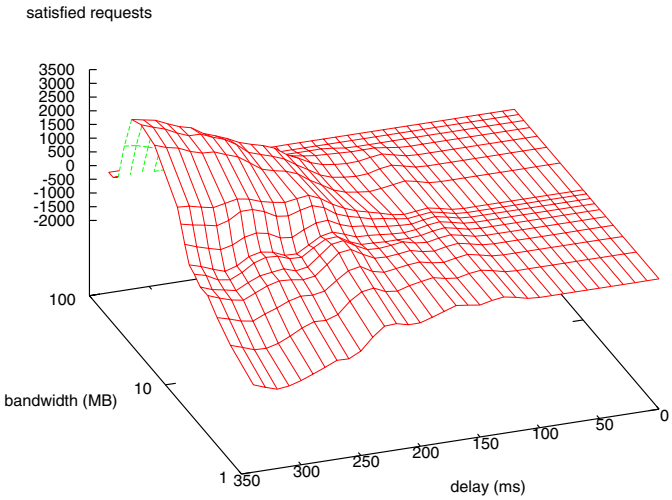

Fig. 8. Back view of the Fig. 7

C. *Two constraints (delay and bandwidth) paths*

For the multi-constrained path search, the most efficient algorithm is our `arc+` algorithm. Fig. 3 shows the number of satisfied requests obtained with the `arc+` algorithm. The studied graph is composed of 20 nodes. The number of satisfied requests is greater to 8,000 for delays greater than 200 ms and bandwidth smaller than 20 MB. The number of satisfied requests falls rapidly for smaller delays and greater bandwidth. There is not any satisfied request for delay smaller than 50 ms because even for one constraint, there is not any path with delay smaller than 50 ms. The results for the graph of size 50 (Fig. 4) are similar but the number of satisfied requests is decreasing faster.

Fig. 5 and 6 present the difference between the number of results found by our algorithm and the one found by the shortest hop algorithm. Our algorithm gives better results for the strong constraint requests. For the request with delay around 150 ms, the number of satisfied requests is 35% greater. For the requests with a demand of small bandwidth, the shortest hop algorithm finds more results. In this case, the behaviors of both the algorithms are close to their behaviors when they are used to satisfy requests with one constraint only. This case is not critical and the difference of the number of requests satisfied between the `shortest hop` algorithm and the four others is smaller.

Fig. 7 and 8 present the same difference for a graph of size 50. For delays smaller than 200 ms, the two algorithms satisfy the same number of requests. For the requests with big delay and little bandwidth, the `shortest hop` algorithm gives better results. The number of satisfied requests is 20% greater. Starting from 3 MB, the number of requests satisfied by our algorithm is greater than the number of requests satisfied by the `shortest hop` one. For the requests with bandwidth

demand greater than 30 MB, the results obtained with our algorithm are significantly better, up to 35%.

These results shows that our algorithm is better than the `shortest hop` one satisfying requests with large bandwidth demand. For small bandwidth demand (from 1 to 3 MB), the algorithms still works well. Notice that these requests are found in a small number of cases.

VI. Conclusion

We studied a problem of inter-domain resource provisioning taking into account multi-constrainst requests. We proposed a ditributed algorithm to find multi-constraint paths which is a *NP*-complete problem. We designed our algorithm in order to reply to the following expectations:

- using local state knownledge only;
- preserving the domains' independence;
- working with well known constraints (delay, bandwidth, lost rate, jitter) or any emerging constraint which would be additive, multiplicative or convex.

Our heuristic algorithm is based on a depth first search with cut mechanisms. The search is made by a probe sent into the network. The probe contains the information needed for fulfilling the search. The nodes use their local information and the probe's contents only. We proposed also four extensions of our algorithm.

The proposed algorithms give very good results for multi-constrained reservation and they still work correctly for one constraint only. The simulation shows that the stronger the constraints are, the better the performences are.

We have to remark that the application of `shortest hop` algorithm which we have taken as a reference for simulations is not realistic in the Internet. The behavior of BGP is more complex and the routes selected by BGP are different from the shortest hop routes. It will be interesting to test the algorithm on real inter-domain network with a BGP mechanism to compare our results.

References

[1] R. Braden, L. Zhang, S. Berson, S. Herzog, and S. Jamin, "RFC 2205 - resource reservation protocol (RSVP) – version 1 functional specification," Standards Track RFC 2205, The Request for Comments, September 1997.

[2] Murali Kodialam and T. V. Lakshman, "Minimum interface routing with applications to MPLS traffic engineering," in *IEEE INFOCOM*, 2000, pp. 376–385.

[3] Subhash Suri, Marcel Waldvogel, Daniel Bauer, and Priyank Ramesh Warkhede, "Profile-based routing and traffic engineering," *Computer Communications*, vol. 26, no. 4, pp. 351–365, 2003.

[4] Su-Wei Tan, Sze-Wei Lee, and Benoit Vaillaint, "Non-greedy minimum interference routing algorithm for bandwidth-guaranteed flows," *Computer Communications Journal*, vol. 25, no. 17, pp. 1640–1652, November 2002.

[5] Hans De Neve and Piet Van Mieghem, "TAMCRA: A tunable accuracy multiple constraints routing algorithm," *Computer Communications*, vol. 23, pp. 667–679, 2000.

[6] Erol Gelenbe, Michael Gellman, Ricardo Lent, Peixiang Liu, and Pu Su, "Autonomous smart routing for network QoS," in *Proceedings of International Conference on Autonomic Computing*. 2004, pp. 232–239, IEEE Computer Society.

[7] Douglas S. Reeves and Hussein F. Salama, "A distributed algorithm for delay-constrained unicast routing," *IEEE/ACM Trans. Netw.*, vol. 8, no. 2, pp. 239–250, 2000.

[8] J. M. Jaffe, "Algorithms for finding paths with multiple constraints," *Networks*, , no. 14, pp. 95–116, 1984.

[9] Turgay Korkmaz and Marwan Krunz, "A randomized algorithm for finding a path subject to multiple QoS requirements," *Computer Networks (Amsterdam, Netherlands: 1999)*, vol. 36, no. 2–3, pp. 251–268, 2001.

[10] Li Xiao, King-Shan Lui, Jun Wang, and Klara Nahrsted, "Qos extension to bgp," in *ICNP '02: Proceedings of the 10th IEEE International Conference on Network Protocols*, Washington, DC, USA, 2002, pp. 100–109, IEEE Computer Society.

[11] Maciej Liśkiewicz, Mitsunori Ogihara, and Seinosuke Toda, "The complexity of counting self-avoiding walks in subgraphs of two-dimensional grids and hypercubes," *Theor. Comput. Sci.*, vol. 304, no. 1-3, pp. 129–156, 2003.

[12] "BRITE, Boston University Representative Internet Topology gEnerator," http://www.cs.bu.edu/brite/.