Noname manuscript No.
(will be inserted by the editor)

Distributed Busacker-Gowen Algorithm for end-to-end QoS pipe negotiation in X-domain networks

Hélia Pouyllau · Stefan Haar

the date of receipt and acceptance should be inserted later

Abstract Multi-media services and other critical multi-site services (e.g. VPN) are becoming mainstream, and require a guaranteed Quality of Service (QoS). Services need to be established across several Autonomous Systems (ASes), often to connect end-users. Thus, provisioning and control of *end-to-end* QoS requirements arise as ones of the main challenges in Inter-AS management. The *contractual approach*, consisting in using Service Level Agreements (SLAs) defined by each crossed AS, allows to negotiate contract chains that satisfy end-to-end requirements. However, establishing such chains by on-demand negotiations does not scale up for large numbers of requests. Hence, we propose a negotiation process to occur before users' requests to establish service are received. The proposed negotiation process results in the selection of aggregated contract chains, called pipes, and a distribution between them. Such a distribution would indicate for each chain of a pipe, the connection flow it may accept. In this paper, we address the pipe negotiation problem as a network flow problem. We also propose a distributed adaptation of an algorithm for network flow problems.

Keywords. End-to-end QoS, Inter-AS, QoS pipes, SLA, negotiation, distributed algorithms, network flow, Busacker-Gowen algorithm.

1 Introduction

When endeavouring to propose on-demand services to remote users on the Internet, Service Providers (SPs) are facing several challenges. One of these challenges consists in guaranteeing end-to-end Quality of Service (QoS) requirements for critical services such as video-conferencing, VPN, etc. Several constraints have to be met due to the distributed nature of the Inter-AS topology on which the Internet is based. Autonomous Systems (ASes) are pairwise connected and may be administrated by different systems (e.g. DiffServ, etc.). In particular, ASes are managed independently of one another.

H. Pouyllau
Alcatel-Lucent Bell Labs, route de VilleJust, 91620 Nozay, France, E-mail: helia.pouyllau@Alcatel-Lucent.fr,

S. Haar
IRISA/INRIA, Campus de Beaulieu, F-35042 Rennes cedex, France,
E-mail: stefan.haar@irisa.fr

The pressure to guarantee QoS between independent partners leads SPs to design QoS contracts (Service Level Agreements, SLAs). ASes allow their end-users to access services through SPs, interfacing with other ASes through a service layer, such as recommended by the IPSPhere Forum [6]. Each QoS class is composed of several QoS parameters (e.g. delay, etc.) whose values are set with respect to the service type (e.g. video-conferencing, etc.). In the case of end-to-end QoS guarantee, a **chain of point-to-point contracts** has to be created whose combination satisfies an initial QoS requirements. We borrow from [24] the expression **QoS budget** for the end-to-end QoS requirements, in order to highlight the accumulation effects that QoS parameters are subject to: the total delay of a contract chain is the sum of contract delays, the total bandwidth is the minimum, etc. Therefore, the client's overall tolerance to QoS weaknesses expressed in the budget is gradually consumed by the successive contributors.

Thus, before establishing a service, a negotiation process has to occur so as to allocate shares of the QoS budget to participating ASes. Such a negotiation process must respect the AS independence and contract privacy. Hence, it has to be executed in a **distributed** manner.

In previous works [12, 18, 20], we proposed two distributed algorithms for this negotiation. Both of these algorithms use the Dynamic Programming principles. The first algorithm, detailed in [12], is a distributed adaptation of a Dynamic Programming algorithm - called Viterbi algorithm; the other one is a distributed version of a 2-sweeps Dynamic Programming algorithm [20] - the Forward-Backward algorithm. Both algorithms optimize a single contract chain in isolation. And, the second one provides the optimal solution for several requests but has lower performances of computing time. In fact, negotiation per request can be quite slow, depending on the number of participating ASes and their number of local contracts.

Furthermore, inter-AS resources takes time to be configured. Thus, for services that need a guaranteed QoS but for small amount resources (services targeting *people at large*, e.g. VoD, voIP, etc.) a negotiation process will not occur at each end-user demand. Moreover, it is desirable to keep the amount of negotiation message exchanges prior to service as low as possible, particularly so for pairs of ASes that are often interconnected. For such cases, it is preferable to pre-negotiate - **offline** - QoS contract chains instead of launching a negotiation each time a service establishment request occurs. Therefore, we addressed in [19] the problem of negotiating *aggregations*, called *pipes*, of QoS contract chains. That is, the negotiation result is a distribution of requests over several different pre-established paths; such a pipe allows to balance the loads over those chains, taking into account the expected number of service requests concerning the same type of service between the same two domains.

However, this first approach for pipe negotiation has some drawbacks. First of all, the algorithm does not have very good runtime performances. On the other hand, the process we proposed does not respect entirely the data condidentiality between ASes. In this article, we describe another approach which consists in considering the pipe negotiation as a network flow problem and adapting a well-known algorithm from that area.

The present article starts by discussing the motivations for studying pipe negotiation, and then presents the formalization of the problem. In section 5, we propose to use a distributed adaptation of the Busacker-Gowen algorithm which had been originally designed for solving minimal cost flow problems. This adapation of the Busacker-Gowen algorithm solves the problem of computing a pipe at a minimal cost (MCP).

2 Motivations and Context

Consider the process of establishing a video conference (Figure 1): an end-user, U, using service at domain, e.g. d^0 who wants to begin a conference with another one in domain, say, d^3 sends a request to the SP of its AS. This request contains the desired service type and end-to-end QoS requirements; the latter depend on the selected service (here: video-conference) and the end-user's choices (e.g. high quality service, medium quality, etc.) which are transformable into real thresholds on QoS parameters.

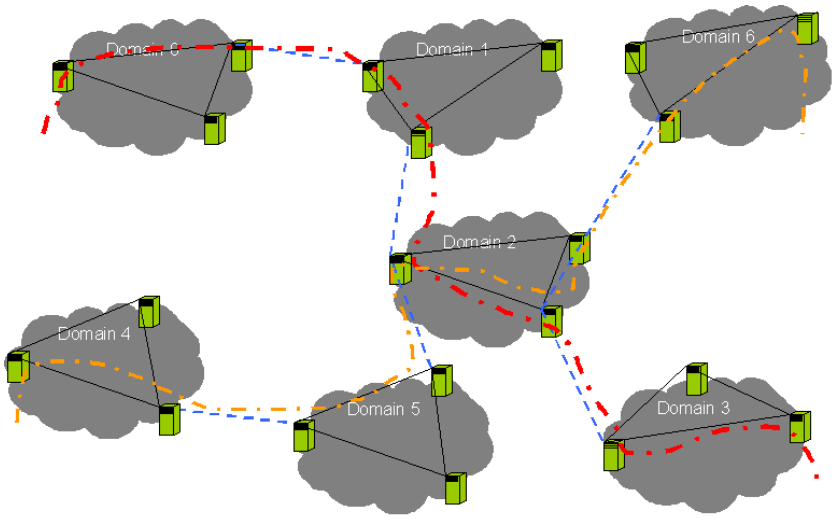

Fig. 1 Inter-AS provisioning.

2.1 QoS guaranteed services

The future of the Internet requires switching from a "best-effort" infrastructure - performed through the Border gateway Protocol (BGP) - to a QoS guaranteed one. Best-effort services will still be needed in the future, but the emergence of high performance services have underlined the limit of the inter-domain routing protocol (BGP). Some authors [25, 11] proposed QoS extensions of BGP. However, these approaches did not manage to overcome two central difficulties: *i*) scalability issues (computing multi-constrained paths is an NP-Complete problem); and *ii*) issues resulting from dynamicity: maintaining QoS-extended routing tables for resource reservations would generate a huge amount of messages.

Other authors and consortiums [16, 1, 4, 6] have recommended alternative approaches, based on the assumption that the inter-AS service establishment question has to be solved at a business level, using QoS contracts (Service Level Agreements, SLAs). Such contracts are then mapped to the actual network resources. However, this mapping process can not be performed in real time since resource configuration procedures are not

yet efficient (consider for instance the establishment of a LSP in the (G)MPLS infrastructure). For these reasons, the target services in the inter-AS context are often applications that need a huge amount of *resources* (e.g. grid services, surgical teleoperation, etc.). Concerning services dedicated to big numbers of *users* (e.g. VoD, voIp, video-conference, etc.) a solution would be to negotiate inter-domain services for a specified number of connections. In this article, we study this option and how it fits into the current intra- and inter- AS context.

2.2 Intra- and Inter-AS assumptions

The network architecture and infrastructure may be different for the different individual ASes(e.g. DiffServ, etc.); as a consequence, the processes for internal provisioning, monitoring, etc. may be incompatible between ASes. In [1], we proposed an architecture for heterogeneous ASes, integrating internal provisioning and monitoring using interoperable technology (e.g. Web Services, JMS, CORBA, etc.). The negotiation approach we propose is independent of the underlying architecture and we do not address the underlying intra-AS problems.

QoS Classes. Locally, we assume each AS, d^i with $i = 0, \ldots, N$, defines a set of *QoS classes*, noted $\mathbb{C}^{\mathbf{i}}$, on each internal link. A QoS class is given by service types and levels for QoS properties. This assumption implies that we consider the QoS class configuration problem as solved; in fact, mechanims exist for admission control [13] (to ensure that the end-user accesses to the resources dedicated to his demand) and QoS class policing [16] (to ensure that each domain can support a specific QoS class). A QoS contract (or QoS class instance), noted $\mathcal{C}$, stipulates levels of QoS for different parameters, in a Service Level Agreement or SLA. We focus on parameters that can be represented by numerical components such as thresholds and probabilities, and allow computational accumulation. Combining the values for different parameters, the relevant parts of an SLA can thus be represented by numerical vectors.

Resources and pricing. QoS classes are attributed to an AS; each AS is subject to **resource constraints**: only a limited number of contracts for different QoS classes can be simultaneously satisfied. So, each class is associated to a numerical capacity indicating the number of contracts that can be allocated for this class. We assume resource consumption is increasing w.r.t. QoS, that is, contracts of superior QoS use more resources. We assume also that a *fixed price* is associated to each class. Obviously, prices are linear to resource consumptions: a class with a low price has also a low consumption. This assumption is valid for all QoS classes which do not include best-effort class. We consider that processes like QoS class negotiation and provisioning would not be used for best-effort trafic which can be routed through domains using BGP.

Inter-AS routing. ASes are pair-wise connected over *adjacency links*. Each AS can be assumed equipped with *routing tables* that indicate, for the target domain, the next hop to be chosen according to the shortest path to target; these tables are pre-established offline, by a BGP-like protocol [22] for instance which provides a main inter-AS route.

2.3 End-to-end QoS: the cumulative effect

We suppose end-users have the possibility to choose between different levels of QoS. These levels (e.g. premium, etc.) correspond to constraints on QoS parameters, and of course to different prices.

When an end-user, noted U^0, of an AS d^0 (compare Figure 1) wants to access a service provided by an AS d^N, U chooses a QoS level which will have to be respected by the future path between two end points, d^0 and d^N. As each AS proposes a set of QoS classes for each type of service, a **chain of neighbor-to-neighbor QoS contracts** has to suit the QoS requirements desired by the end-user. These QoS contracts have the same mathematical form as QoS requirements: they are composed of several QoS parameters (e.g. delay, jitter etc.) which are subject to cumulative effects as explained above.

In [12], we have described a negotiation process which handles the cumulation effects of QoS contracts and select the best QoS commitment path w.r.t a criterion like the sum of prices. The following aspects were crucial in developing the negotiation procedure.

Nesting. Negotiation can only occur between two peers at a time, and neither be centralized nor based on information about distant sites. That is, neighboring sites have to negotiate middle-to-end QoS commitments as contracts between the neighbors as illustrated by figure 2: if AS d^0 is to be connected to AS d^N through ASes $d^1, \ldots, d^N$, then AS d^i negotiates with d^{i+1} a commitment on the part of AS d^{i+1} toward AS d^i, in which d^{i+1} is responsible for the QoS on the subchain $d^{i+1}, \ldots, d^N$.

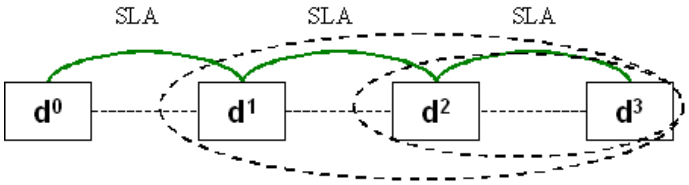

Fig. 2 Inter-AS nested negotiation.

Universality w.r.t. types of composition. QoS budgets (or QoS contracts) aggregate several parameters which may have different mathematical properties. We speak of *composition*, noted $\oplus$, for the action of "adding" two contracts, and of *decomposition*, noted $\ominus$, for that of "subtracting" a contract from a QoS budget. Depending on the parameter type, $\oplus$ must be interpreted differently, e.g. as *sum* for additive parameters such as delay, *product* for multiplicative ones like availability, *min* for bandwidth, *convolution* for probability laws, and so on; the negotiation protocol needs to be independent of the mathematical type of composition.

Multi-dimensionality. Optimization of contracts is to be performed with respect to **several QoS constraints** simultaneously (e.g. respecting low delay AND low jitter). That is, the protocol needs to handle *vectors of thresholds* on different QoS parameters. Note that, in the light of the previous requirement, we may have different composition rules in different components, so the genericity of vector composition is crucial. Furthermore, the selected aggregate should be as optimal as possible following

a selection criterion (e.g. the total prices of the chains, the global revenue generated for the ASes, etc.).

2.4 Related work

Negotiation approaches. The authors of [26] consider the control admission of users in case of single contract chains and also for pipes. A Bandwidth Broker is in charge of defining reservation to make in order to respect QoS constraints. The Bandwidth Broker also distribute users between the reserved chains. The solution they proposed is therefore more oriented towards control admission although they advise to provide some distributed process for reservation.

In [14], authors study the definition of a flow in a network using MPLS (Multi-Protocol Label Switching [23]) [23]. This routing protocol uses labels to route packets inside the AS network. The authors of [14] address the problem of finding paths satisfying a QoS class in a local network and then defining the maximal flow of connections associated to these paths. They solve both problems separately: a first algorithm defines the paths which meet the QoS constraints; and, a linear program solves the remaining problem.

The authors of [5] search to specify the maximal bandwidth between two end-points for a given service type crossing several domains. They assume QoS classes are defined by each crossed AS for the given service type. Thus, the problem they address consists in selecting a class on each AS such that : *i*) a bandwidth constraint is respected; *ii*) the selected classes maximize the AS revenues. Their approach follows micro-economical considerations to guarantee end-to-end QoS. QoS class pricing is used in order to avoid congestions. Although their work does not consider end-to-end QoS requirements, we plan in future work to use their work on micro-economical questions.

The authors of [17] study the same problem as the authors of [14]: how to define pipes in a multi-service context assuming routing table of the AS are administrated using MPLS. They suggest to use first an enumeration algorithm to find paths satisfying the QoS constraints; and a genetical algorithm determines flows in a centralized manner. They compare the performances of their solution to a linear program. Their algorithms seem to provide solutions close to the optimum. However, in term of execution time their process appears to be slower than the linear program they compare it to. We suppose this is due to the enumeration which precede the genetical algorithm.

The work [11] and the MESCAL project [10] address the same problem as we do: how to find a flow on contract chains that optimize a global cost function and respect a required end-to-end QoS. Their solution is close to the solution of [17]: an enumeration of the admissible contract chains precedes the execution of a genetical algorithm; both are performed centrally. Thus, their algorithms, besides the fact that they use an enumeration (which implies low performances), are not adapted to the Inter-AS context: their processes are *centralized* so, they do not respect the AS independence.

Other works on end-to-end QoS provisioning [7, 15, 8] do not propose algorithms for finding and reserving optimal chains of contracts. The approaches of [7, 15, 8] are constraint solving approaches.

Previous work. As mentioned in the introduction, we studied in previous work the negotiation of a simple contract chain satisfying end-to-end QoS requirements. Our approach was inspired by the use of the Dynamic Programming principles. In [12], we proposed a distributed adaptation of the Viterbi algorithm. This algorithm

solves the simple chain negotiation problem for an individual request. It proceeds in two sweeps: *i*) from the source to the target domain, each domain computes all the possible remaining QoS budgets by decomposing the received budgets using its local QoS classes - this set of output budgets is pruned w.r.t. their costs (there lies the Dynamic Programming optimization); output budgets and their associated costs are transmitted to the next domain; then, *ii*) the target domain selects the best budget w.r.t. its costs and transmits this selection' budget value backward along the chain until it reaches the source domain.

In order to reach the optimal solution for several simple chain requests, we designed a distributed version of the Forward-Backward algorithm [2] which we described in [18, 20]. This algorithm performs the first sweep of the Viterbi algorithm twice (from the source to the target and from the target to the source). Therefore, each domain knows the possible budgets from the source and from the target. Using this information, it can proceed to the selection of the local adapted QoS class. Considering several requests, a domain can thus optimizes its class allocations - which may imply to inform its partners about changes.

Because of the low performances of negotiation on request, we studied the possibility of pre-negotiating aggregates of contract chains [19]. In [19], we proposed an algorithmic solution that computes an aggregated of QoS contract chains w.r.t. QoS requirements and provides a repartition of a required number of connections over these chains. In this article, we aim to describe a new algorithmic approach with improved runtime performances. The purpose of this solution would be to replace the algorithm proposed in [12, 18, 20] when negotiating services targeting people at large.

3 The pipe negotiation problem

3.1 Pipes: collections of End-to-end QoS chains

While the above negotiation principles were established in view of single request negotiation, they remain valid in the case of pipes; the difference is that, instead of a single *optimal* path, we need to find *all* admissible paths. A pipe request is initiated by an AS or a service retailer which wants to negotiate several contract paths for later use and distribute the load, i. e. the expected total number of requests to be treated, among those paths. The purpose of the pipe negotiation is twofold: given a pipe request, find *all* the chains of contracts, between source and target domains, that respect the initial QoS budget $\mathcal{Q}^0$ required by the request.

Let λ be the (unknown) total number of connections to be handled. The number of connections could be estimated using learning mecanisms on customer demand. Another way to evaluate customer demand is to consider specific systems with price differenciation or biding. Such mechanisms are outside the scope of this article. We denote by $\rho_l \in [0, 1]$ the portion of the connections allocated by the pipe to the path l; that is, the average absolute number of requests allocated to l is $\rho_l \cdot \lambda$. Of course, the sum of the ρ_l has to be 1.

Pipe request. Therefore, a request for negotiating a pipe (or pipe request) is constituted by *i)* a QoS budget $\mathcal{Q}^0$, *ii)* an integer λ representing the estimated number of service requests, *iii)* the initiator or source AS d^0, and *iv)* the target AS d^N.

3.2 Example

Fig.3 illustrates an example of pipe. The source AS d^0 initiates a pipe request with a budget requiring a delay under 40ms and an availability of 80% for 100 service requests. Assume that the negotiation process results in a pipe composed of *two* paths. The first path consists in the selection of contracts (15ms, 95%) in AS d^1, (15ms, 93%) in d^i and (5ms, 95%) in d^N, which leads to a total price of $700(= 300 + 100 + 300)$. The second path is identical except in AS d^1 and d^i, where contracts (25ms, 93%) and (10ms, 95%) are selected.

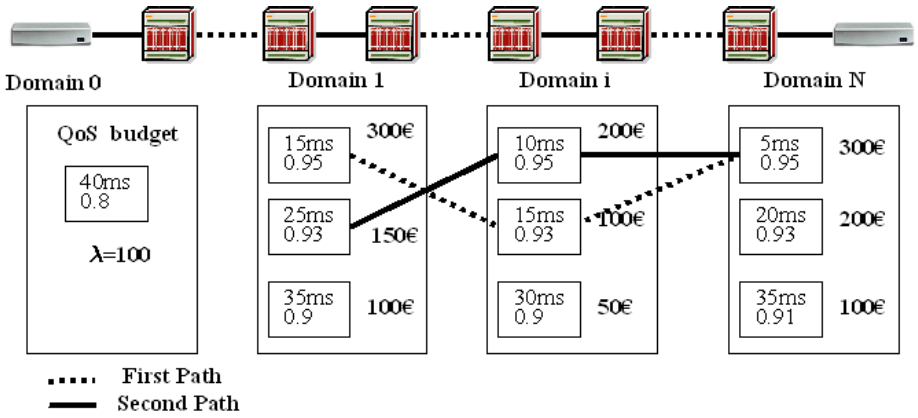

Fig. 3 Example of pipe.

From the point of view of the AS d^0, it would be optimal to select the second path only (i.e. $\rho_2 = 1$), whose sum of prices is equal to $650(= 150 + 200 + 300)$. However, physical constraints, together with traffic from other requests, have already heavily loaded AS d^i, such that it can only provide the contract (10ms, 95%) for at most 75 service connections. The load ratios ρ need to be selected at the *path* level, i.e. be feasible for all ASes jointly. Here, we suppose that due to capacity constraints on each AS the optimal distribution is: 25% of the connection on the first path and 75% on the second one. Note that all domains, including d^N, have to treat two paths, even if the local contracts requested by different paths are the same on d^N. For instance, the resulting pipe on d^N consists of the pairs $< (5, 0.95), 0.25 >$ and $< (5, 0.95), 0.75 >$.

4 Problem statement

In this article, we give a **fully distributed** solution for the pre-negotiation of contract chain aggregates (or pipes). The problem is related to network flow problems, which have been intensively studied in the past, leading to well-known solutions.

Network flow problems. A network is an oriented graph to whose edges are associated flow capacities and eventually other attributes. The network flow problem consists in finding a flow - of capacity - between a source and a target node that respects the edges capacities. Therefore, this original problem has been studied in various forms

and led to various algorithms among those is the Busacker-Gowen algorithm [3]. This algorithm solves the problem of finding a flow of minimal cost respecting a required capacity. The common point with the pipe negotiation problem is the idea of splitting connections between different paths in a network. However, some differences remain:

- the routing assumptions are different. In the context of inter-AS negotiation, we consider the exploration of one route - provided by BGP tables - between a source and a target. In network flow algorithms, every routes are considered. However, in the inter-AS context, each route leads to several possible contract chains since each AS defines locally a set of QoS classes between egress points;
- the existing network flow problems do not consider any constraints on the flows. The only constraints appearing in some problems are lower or upper bounds of capacities on edges. In the case of pipe negotiation, contract chains must respect a QoS budget and local capacity constraints. This point is crucial for the adaptation of the existing algorithms.

Pipe negotiation as a network flow problem. The shortest path between an AS d^s and an AS d^t, as established by the BGP protocol, is noted $\pi_{s\to t}$. To determine whether or not a path may satisfy the QoS budget $\mathcal{Q}^0$ of a pipe request, we give a definition of a multi-AS network that integrates QoS attributes. We call such a network "an AS contract network" because nodes represent ASes, and edges possible contracts.

Definition 1 (AS Contract network) A multi-AS network is an oriented graph $G = (\mathcal{D},\mathcal{U})$ where each node $d^i \in \mathcal{D}$ is an AS, and each edge $u = (i,j) \in U$ is a QoS class which can be provided by the d^j to d^i. To each edge $u = (i,j) \in U$ is associated

- $\mathcal{C}_u^j$: a QoS class, defined by AS d^j, such that $\mathcal{C}_u^j \in \mathbb{C}^\mathbf{j}$,
- Cap_u: the maximal capacity of $\mathcal{C}_u^j$, i.e. the maximal number of contracts this class can be instanced by,
- p_u: the price of $\mathcal{C}_u^j$.

Consider the example in figure 4, showing one isolated route. ASes are represented by nodes and QoS classes by edges. The entering edges of a node are its QoS classes. Fig. 4 illustrates an example where each AS has defined 3 classes. This number has been arbitrary chosen. However, it is realistic to have such a low number, in the sense that some studies recommend a certain predictability of the system for the users: the number of classes must be limited and if prices are dynamically computed - regarding the demand, this dynamicity should occur at a low frequency [21]. Thus, as QoS classes

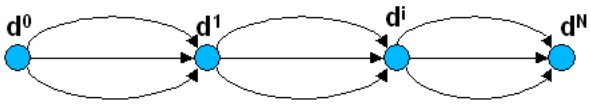

Fig. 4 AS graph.

are associated to the edges of the AS network, we give a formal definition of a contract chain using the previous definition 1.

Definition 2 (Contract chain) A contract chain $\mathcal{P}$ is a tuple

$$\mathcal{P} \triangleq (\pi_{s\to t}, (\mathcal{C}_u)_{u\in\pi}, p, \min_{u\in\pi} Cap_u)$$

where:

- $\pi_{s\to t}$ is a path between source node d^s and target node d^t,
- $(\mathcal{C}_u)_{u\in\pi}$ is the collection QoS classes associated to the edges involved in the path π. Their composition must satisfy $\bigoplus_{u\in\pi}\mathcal{C}_u \leq \mathcal{Q}^0$,
- p is the cost for allocating resources of $\mathcal{P}$, for instance $p = \sum_{u\in\pi} p_u$, the sum of QoS class prices of π,
- $\min_{u\in\pi} Cap_u$ is the maximal number of connections a path π may hold, it is therefore the minimum of the QoS class capacities associated to π's edges.

With this definition, we propose to adapt the existing network flow algorithms to solve the pipe negotiation problem. To respect ASes'independence, any pipe negotiation problem has to be solved in a **distributive manner**. Thus, we propose a **distributed version of the Busacker-Gowen algorithm** to solve the problem of finding the **minimal cost pipe problem (MCP)**.

5 Minimal cost pipe problem (MCP)

The minimal network flow problem consists in finding a flow of a required capacity at a minimal cost. This problem is close to the minimal cost pipe problem (MCP) we want to address: In the MCP problem, the QoS constraints make the problem more complex than the minimal network flow problem.

5.1 Formulation of the problem

The objective is to determine the value of the φ_u variables. Each φ_u indicates the number of possible contracts (and therefore the number of connections) available for a QoS class on edge $u \in \pi_{s\to t}$. These variables are positive integers. Each φ_u of an edge $u \in \pi_{s\to t}$ belonging to an admissible contract chain satisfies $0 \leq \varphi_u \leq Cap_u$. This constraint is expressed through the equation (4).

We strive to minimize the cost of a pipe. The cost function (1) is expressed through the sum of prices of the classes involved in chains satisfying the QoS requirements (2). The price, p_u, of a class is associated to the edge $u \in pi_{s\to t}$ (cf. Definition 1). Thus, the total cost of a pipe is the product of edge capacities and prices.

The equation (3) conveys that the sum, denoted φ, of the flows associated to contract chains satisfying the constraints (2), is bounded by an integer number λ of connections. This constraint is important since it will provide a termination condition for the algorithm.

$$\min_{\varphi} \sum_{u\in\pi_{s\to t}} \varphi_u.p_u \tag{1}$$

$$\text{subject to, } \bigoplus_{u\in\pi_{s\to t}} \mathcal{C}_u \leq \mathcal{Q}^0 \tag{2}$$

$$\text{and, } \varphi \leq \lambda \tag{3}$$

$$\text{and, } 0 \leq \varphi_u \leq Cap_u \text{ with } \varphi_u \in \mathbb{N} \tag{4}$$

5.2 A distributed version of the Busacker-Gowen algorithm

The Busacker-Gowen algorithm [3] consists in iteratively searching for flows in a directed graph such that the global cost is minimized subject to capacity constraints. The algorithm terminates when the required global flow λ is reached; it proceeds as follows:

1. Search for a path $\pi_{s\to t}$, between the source and target node, with the minimal cost. The capacity of this path is equal to the minimum of the capacities of the edge composing the path. Such a path may be found using a shortest path algorithm such as the Bellman or Dantzig algorithms [9].
2. Update the graph, for each edge belonging to the path, $u \in \pi_{s\to t}$:
   (a) Add an edge u^- in the opposite sense or update it by setting a capacity equal to the difference between the capacity of u^+ and the minimum of the capacity of the edges belonging to $\pi_{s\to t}$.
   (b) Set the capacity of u^+ to the difference between the minimum of the edge capacities of path $\pi_{s\to t}$ and the current capacity of u^+.
3. Return to (1).

To adapt this algorithm to the MCP, we have first to modify slightly the algorithm for the search of a path between the source and target node. In the pipe negotiation problem, this path is constrained by the end-to-end QoS requirements of the source domain. Moreover, the algorithm has to operate in a distributed manner to respect the AS independence and the contract privacy.

Distributed Busacker-Gowen algorithm. The objective of the algorithm 1 is to determine the value of a vector φ whose size is $\mid U \mid$. The elements of vector φ represent the capacity of each edge. Each iteration k of the algorithm begins with the search of a path between the source and target node that respects the end-to-end QoS requirements. To determine this constraint path, we propose to use an adapted version - which we presented in [12] - of the Viterbi algorithm. This algorithm applies the Dynamic Programming principles to compute the contract chain that respects an end-to-end QoS budget and optimizes a cost function. The initial QoS budget is transmitted to the next AS in the route - provided by routing tables; that AS then determines **output QoS budgets and associated costs** by decomposing the input budget with the local QoS classes. Output budgets are computed recursively along the path, until the target. The target AS have then all the necessary information to select locally the contract chain that optimizes the cost function. Note that only QoS budgets are communicated, and thus contract privacy is respected.

Adaptation of the Viterbi algorithm. In the case of the MCP, the cost function (1) is defined by the sum of the possible QoS class prices. The Viterbi algorithm allows to compute the contract chains which respect to the QoS budget constraints. In [12], we assumed that the last node - the target AS - executing the adapted Viterbi algorithm is in charge of selecting the contract chain at the least price, noted $\mathcal{P}^*$. Thus, the selection criterion of the adapted Viterbi is the same as the one necessary in the adaptation of the Busacker-Gowen algorithm: the last involved AS selects the contract chain satsfying the QoS budget at the least price. Then, at the end of an iteration k, the variables φ_u^k - representing the consumed capacity of an edge $u \in U$ - are updated w.r.t. the remaining capacities of the ASes involved in the last computed path, $\mathcal{P}^*$. This update is processed backwardly in a distributed manner, each node computing the remaining

capacity of a QoS class from the resource consumption of the selected contract chain, $\mathcal{P}^*$. Therefore, the residual network is built and updated in a distributed manner.

As the Viterbi algorithm would be performed k times, we recommend some changes to store resulting computations of constraint solving. In this way, at each iteration k the Viterbi algorithm would check the residual capacity of the involved QoS classes. The termination of the algorithm 1 is guaranteed in all cases: impossibility of satisfying the initial QoS budget, selling out of capacities, or satisfaction of the required number of connections.

Algorithm 1 Distributed Busacker-Gowen

$\varphi = (0, 0, ..., 0)$, k=1, ϵ=0
$\forall\ i \in \pi_{s \to t}$, negotiateViterbi$_i$ builds $\mathbb{P}$ //$\mathbb{P}$, *set of admissible chains optimizing* (1)
while $\varphi_0^k + \epsilon \leq \lambda$ or $\mathbb{P} \neq$ nil **do**
  $\mathcal{P}^*$ such that $\min_{\mathcal{P} \in \mathbb{P}}(\sum_{u \in \pi_{\mathcal{P}}} p_u)$ //$\mathcal{P}^*$, *the best path is selected*
  $\forall\ u \in \pi^*_{\mathcal{P}}$, $\varphi_u^{k+1} \leftarrow \varphi_u^k + \epsilon$
  $k \leftarrow k + 1$
  $\epsilon = \min_{u \in \pi^*_{\mathcal{P}}} Cap_u$
  for $i \in \pi_{s \to t}$ **do**
    if $u = (*, i) \in \pi_{\mathcal{P}^*}$ and $0 \leq \varphi_u^k \leq Cap_u$ **then**
      $u^+ = (*, i)$ such that $p_{u^+} = p_u$ and $Cap_{u^+} \leftarrow Cap_u - \varphi_u^k$
    end if
    if $u = (*, i) \in \pi_{\mathcal{P}^*}$ et $0 < \varphi_u^k \leq Cap_u$ **then**
      $u^- = (*, i)$ such that $p_{u^-} = p_u$ and $Cap_{u^-} \leftarrow \varphi_u^k$
    end if
  end for
  $\forall\ i \in \pi_{s \to t}$, negotiateViterbi$_i$ builds $\mathbb{P}$
end while

Complexity. The complexity of the centralized Busacker-Gowen algorithm is $\mathcal{O}(N^4)$ [9]. However, a crucil component of this complexity is that of the shortest path algorithm. In our case, the complexity of the Viterbi algorithm is bounded by $\prod_{i=1}^{N} \mid \mathbb{C}^{\mathbf{i}} \mid$. Thus, for k number of iterations, the complexity of the distributed Busacker-Gowen algorithm we propose is bounded $(k+1).\prod_{i=1}^{N} \mid \mathbb{C}^{\mathbf{i}} \mid$. Indeed, one more execution is done before the incremental part of the process, this explains the $k+1$. The number of iterations is also function of the total required flow λ and the capacities of admissible chains.

5.3 Performances

In this section, we present the evaluation of the runtime performances of negotiation algorithms. The purpose of these experiments is to check the scalability of the algorithms we designed and to compare their performances to a benchmark algorithm. The algorithms we evaluated are the followings:

- a *Branch and Bound* algorithm: this type of algorithms are well-known in the combinatorial optimization area. However, their runtime is bounded by an exponential function and they have to be executed in a centralized manner;
- the distributed adaptation of the *Forward-Backward* algorithm: we proposed this algorithm in [20] and use it for solving the QoS pipe negotiation problem in [19];
- the distributed adaptation of the *Busacker-Gowen* algorithm described above.

We tested these algorithms for the negotiation of one request for a contract that requires to cross 5 domains. We observed their individual time of execution as the number of QoS classes per domain varied (from 3 to 180). For the distributed algorithms considered, runtime is measured from the reception of the request to the resulting QoS pipe answer. Therefore, the delay between intermediate messages are taken into account. The simulation was done in the worst case scenario: all the QoS classes of the 5 domains lead to combinations that respect the QoS requirements of the demand.

Figure 5 illustrates the runtime curves of the algorithms : on the Y-axis appears the time expressed in seconds and on the X axis the number of QoS classes per domain. Figure 5 shows that the runtime of all algorithms increase following the number of QoS classes per domain. At approximately 135 classes per domain, the distrbuted adapatation of the Forward-backward algorithm exhibits stronger runtime performances than a classical Branch and Bound. We underline also that the distributed adaption of the Busacker-Gowen algorithm we proposed seems to be the most efficient algorithm.

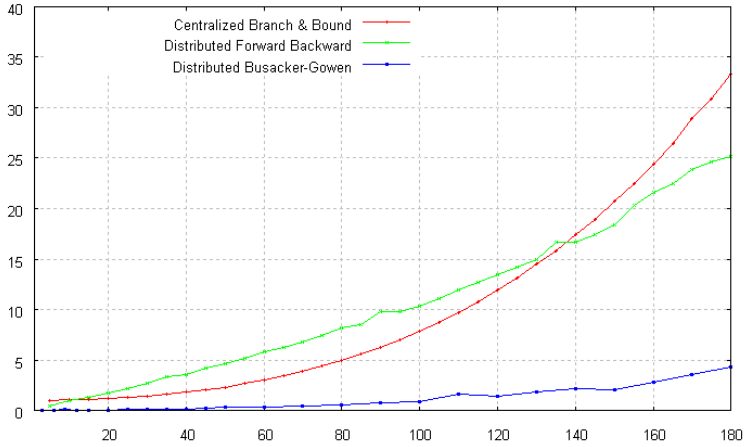

Fig. 5 Performances of the algorithms solving the simple chain negotiation problem.

6 Conclusion

In this article, we formalized the end-to-end QoS pipe negotiation problem as a network flow problem. The adaptation of Busacker-Gowen algorithm we designed is entirely distributed and therefore well-adapted to the multi-AS context. Moreover, the information exchanged during the distributed Busacker-Gowen algorithm does not allow to deduce any private data (e.g. capacity of resources). The experiment results confirm the good performances of this distributed algorithm.

We believe that pipe negotiation is an important help for end-to-end QoS contract chain negotiation. As the negotiation of simple end-to-end chains turns out to be costly, the negotiation of pipes can be executed offline and still offering an optimal aggregate of contract chains. Although we did not compare our approach to a heuristic algorithm,

we believe that the Busacker-Gowen algorithm performances in a simulation benchmark are competitive.

Moreover, if we compare our approach to the heuristic approaches existing in the literature (cf. Section 2.4), we do not use any enumeration of the possible contract chains. Therefore, to compare our approach to a heuristic one, we would have to redefine entirely such a heuristic approach because, in our opinion, the existing ones are perfectible.

Obviously, the distributed version Busacker-Gowen algorithm which answers particularly well to the initial problem, has to be evaluated in a more realistic context, where they are executed at the service layer level of several domains each of one having a control plane able to instantiate and reserve the QoS paths. Such experiments are planned as future work.

Several other perspectives are opened by this work. The first one is to create self-healing mechanisms for pipes. Like simple chains, aggregated chains may be subject to contract violations; renegotiation mechanisms and the violations cases remain to be studied. Another line of research consists in finding an optimal solution in the case of several pipe requests. The work achieved in this article considers the optimization for an individual pipe request. Finally, the examination of the economical aspects of pipe and simple chain negotiations emerges as a relevant future work; in particular, economical inter-provider mechanisms for ensuring good use of pipes have to be found, e.g. treatment of contract penalties, etc.

References

1. A. Aghasaryan, S. Piekarec, H. Pouyllau, S. Haar, E. Fabre, L. Ciarletta, N. Mbarek, and E. Moreau. Multi-domain self aware management : Negotiation and monitoring. In *IEEE International Conference on Telecommunications*, 2006.
2. Leonard E. Baum. An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes. In Oved Shisha, editor, *Inequalities III: Proceedings of the Third Symposium on Inequalities*, pages 1–8. Academic Press, 1972.
3. R. G. Busacker and P. J. Gowen. A procedure for determining a family of minimal-cost network flow patterns. Technical report, Johns Hopkins University, 1961.
4. Richard Douville, Jean-Louis Le Roux, Jean-Louis Rougier, and Stefano Secci. A service plane over the pce architecture for automatic multi-domain connection-oriented services. *IEEE Communication Magazine*, June 2008.
5. Zhenhai Duan, Zhi-Li Zhang, and Yiwei Thomas Hou. Service overlay networks: SLAs, QoS, and bandwidth provisioning. *IEEE/ACM Trans. Netw.*, 11(6):870–883, 2003.
6. IpSphere Forum. www.ipsphereforum.org.
7. Errin W. Fulp and Douglas S. Reeves. Optimal provisioning and pricing of differentiated services using QoS class promotion. In *GI Jahrestagung (1)*, pages 144–150, 2001.
8. S. Giordano and al. Advanced QoS provisioning in IP networks: the european premium IP projects. *Communications Magazine, IEEE*, 41:30– 36, Janvier 2003.
9. Michel Gondran and Michel Minoux. *Graphes et algorithmes, 3e édition revue et augmentée*. Eyrolles, 1995.
10. M.P. Horwarth and al. Provisioning for interdomain quality of service: The MESCAL approach. *IEEE Communications Magazine*, 2005.
11. Michael P. Howarth and al. End-to-end quality of service provisioning through inter-provider traffic engineering. *Computer Communications*, 29:683–702, 2006.
12. H.Pouyllau, L.Ciarletta, A. Aghasaryan, and S. Haar. X-domain QoS budget negotiation using dynamic programming. In *IEEE Advanced International Conference on Telecommunications (AICT)*, 2006.
13. S. Lima, A. Santos P. Carvalho, and V. Freitas. A distributed admission control model for CoS networks using QoS and SLS monitoring. In *IEEE 2003 International Conference on Comunications (ICC 2003)*, 2003.

14. Debasis Mitra. Techniques for traffic engineering of multiservice, multipriority networks, 2001.
15. E. Mykoniati and al. Admission control for providing QoS in DiffServ IP networks: the TEQUILA approach, 2003.
16. T.M.T. Nguyen, N. Boukhatem, and G. Pujolle. COPS-SLS usage for dynamic policy-based QoS management over heterogeneous ip networks. *IEEE Network*, 17(3):44–50, 2003.
17. V. Pasias and d R.C. Papademetriou D.A. Karras a. Traffic engineering in multi-service networks comparing genetic and simulated annealing optimization techniques. In *IEEE International Joint Conference on Neural Networks*, 2004.
18. Hélia Pouyllau and Stefan Haar. Distributed end-to-end qos contract negotiation. In *International Conference on Autonomous Infrastructure, Management and Security (AIMS)*, 2007.
19. Hélia Pouyllau and Stefan Haar. End-to-end QoS of X-domain pipes. In *IEEE International Conference on Heterogeneous Networking for Quality, Reliability, Security and Robustness (QShine)*, 2007.
20. Hélia Pouyllau and Stefan Haar. A protocol for QoS contract negotiation and its implementation using Web Services. In *IEEE International Conference on Web Services, USA*, 2007.
21. Peter Reichl, David Hausheer, and Burkhard Stiller. The cumulus pricing model as an adaptive framework for feasible, efficient, and user-friendly tariffing of internet services. *Comput. Networks*, 43(1):3–24, 2003.
22. Y. Rekhter and T. Li. A border gateway protocol 4 (BGP-4) rfc 1771. RFC Editor, 1995.
23. E. C. Rosen, A. Viswanathan, and R. Callon. Multiprotocol label switching architecture. RFC 3031, 2001.
24. Telecommunications and Internet Protocol Harmonization Over Networks (TIPHON). *Part3: Signalling and Control of end-to-end Quality Of Service (QoS)*. ETSI, 2002.
25. Li Xiao, Jun Wang, King-Shan Lui, and K. Nahrstedt. Advertising interdomain QoS routing information. *IEEE Journal on Selected Areas in Communications*, 22:1949– 1964, 2004.
26. Zhi-Li Zhang. Decoupling QoS control from core routers: A novel bandwidth broker architecture for scalable support of guaranteed services. In *SIGCOMM*, pages 71–83, 2000.