Inductive Routing System based on QoS Bandwidth Optimization

Said Hoceini and Abdelhamid Mellouk

Image, Signal and Intelligent Systems Lab-LISSI
Dept. of Networks and Telecoms, IUT C/V
University of Paris Est Creteil (UPEC)
122, Rue Paul Armandot 94400 Vitry sur Seine, FRANCE.
Email: {hoceini, mellouk}@u-pec.fr

Abstract. — Computing constrained shortest paths is fundamental to some important network functions such as QoS routing and traffic engineering. This paper introduces a polynomial time approximation Quality of Service (QoS) routing algorithm and constructs dynamic state-dependent routing policies. The proposed algorithm uses a bio-inspired approach based on the trial/error paradigm combined with swarm adaptive approaches to optimize three QoS different criteria: static cumulative cost path, dynamic residual bandwidth, and end-to-end delay. The approach uses a model that combines both a stochastic planned pre-navigation for the exploration phase and a deterministic approach for the backward phase. In this paper, we adopt the unified framework of online learning to consider a global cost function. Numerical results obtained with OPNET simulator for different levels of traffic’s load show that the new added module improves clearly performances of our earlier KOQRA.

Index Terms— Quality of Service based Routing, Multi Path Routing, QoS Multimedia constraints, Reinforcement Learning, Residual Bandwidth, Load balancing Routing.

I. INTRODUCTION

The demand for real time and quality of services (QoS) in the network increases as Internet expands. However, the service level is sensitive to the characteristics of network transmission, such as delay, delay jitter, bandwidth, packet loss rate, and cost. Therefore, QoS measures should be introduced to the network so that quality of real time services can be guaranteed.

The goal of QoS routing is to find a network path which satisfies the given constraints and to simultaneously optimize the resource utilization [1]. The integration of QoS parameters increases the complexity of current routing algorithms. In fact, the problem of determining a QoS route that satisfies two or more constraints (for example, delay and cost) is known to be NP-complete [2]. One major difficulty is that the time required to solve the Multi-Constrained Optimal path problem exactly cannot be upper-bounded by a polynomial function. Hence much of the focus over the last few years has been on the development of pseudo-polynomial time algorithms, heuristics, and approximation algorithms for multi-constrained QoS paths [3].

The use of bio-inspired models is one possibility of resolve these problems. Currently, we use bio-inspired models in different aspects in telecommunication networks such as fault tolerance [4], QoS routing [5] [6], self organization [7], Network Security [8], Autonomic Communications [9].

In this paper, we focus our attention in bio-inspired QoS routing policies. Basically, these algorithms selects routes based on flow QoS requirements and network resource availability.

After presenting the related works in section II, we present in next section our algorithmic approach, called AVerageBandWidth (AV-BW) Delay Q-Routing, which allows treating the problems of adaptive QoS-routing in telecommunication networks with irregular traffic. This algorithm is based on the earlier developed K-Optimal path Q-Routing Algorithm (KOQRA) [6], where we add a third criterion regarding residual bandwidth. The performances of this algorithm are evaluated experimentally with OPNET simulator in section IV and compared results with the earlier KOQRA. In section V, we make some concluding remarks and highlight some future areas of work.

II. RELATED WORK

Recently, several studies have been conducted on QoS routing algorithms which integrate the QoS requirements problematic for the routing algorithm. Most efforts have investigated the use approaches based on artificial neuronal intelligence together with biologically inspired techniques such as reinforcement learning and genetic algorithms, to control network behavior in real-time so as to provide users with the QoS they need, and to improve network robustness and resilience. We note the following approaches that have been recently proposed: Routing approach [10], DRQRouting which combines Q-Routing with Dual Reinforcement Learning [11], Cognitive Packet Networks (CPN) [12] based on random neural networks, Swarm Ant Colony Optimization (AntNet) [5] inspired by dynamics of how ant colonies learn the shortest route to food source using very little state and computation, AntHocNet [13] used in mobile ad hoc networks and combines reactive route setup with proactive route probing, maintenance and improvement, BeeAdHoc [14] inspired by the foraging principles of honey bees and KOQRA [6] [15]. This latter is an adaptive routing algorithm that improves the Quality of Service in terms of cost link path and end-to-end delay. This algorithm is based on a Dijkstra multipath routing approach combined with the Q-routing reinforcement learning. The learning algorithm is based on founding K best paths in terms of cumulative link cost and the optimization of the average delivery time on these paths. A load balancing policy depending on a dynamical traffic path probability distribution function is also introduced.
We focus our attention in this paper on developing a whole system capable to take into account static and dynamic QoS criteria in order to optimize a global cost function. The system is based on bio-inspired approaches using reinforcement learning with continuous learning dynamic network parameters. The originality of our approach is based on the fact that our system is capable to take into account the dynamics of the network where no model of the network dynamics is assumed initially. Our approach samples, estimates, and builds the model of pertinent aspects of the environment which is very important in heterogeneous networks.

III. PROPOSED APPROACH: AV-BW DELAY Q-ROUTING.

In the goal to use many criteria in routing decision, we propose a bio-inspired algorithmic approach, called AV-BW Delay Q-Routing. This algorithm is based on K-Optimal path Q-Routing Algorithm (KOQRA) [6], where the objective for performance evaluation is the optimizing three criteria: a static one (the cost cumulative paths), and two dynamic ones (the residual bandwidth and end-to-end delay).

The problems remain with the optimization of the multiple criteria simultaneously. In fact, the complexity becomes very high. To solve this problem, we have used a lexicographic heuristic based on the importance of the metrics. This system is based on two stages (Fig 1.), each one optimizing a local cost function.

\[
\text{Stage 1: Selecting candidate paths based on static criteria: the set of candidate paths which is a subset of all available paths between a source-destination pair of considered routers is selected based on a cost function. This latter combined all QoS static criteria, such as link cost, hop count, error ratio, etc.}
\]

\[
\text{Stage 2: Load balancing traffic based on dynamic criteria: split traffic among a subset of all candidate paths issued by a first stage. The proportional value of each qualified path depends on the evaluation of dynamic criteria such as the residual bandwidth, the end-to-end delay, and so on. This evaluation requires the definition of a real-time cost function based on considered dynamic criteria. For this, we have used a bio-inspired approach based on RL methods.}
\]

The system is based on different modules, each one optimizing a local cost function. For this, we propose a unified formalism based on [22] to see the convergence of our approach. Continuous learning in our system amounts in changing the parameters of the network model using an adaptive update rule whose general formulation is:

\[
w(n) = w(n-1) - \epsilon(n) F(x(n), w(n-1))
\]

Where \(w(n)\) denotes the parameters of the whole system at time \(n\), \(x(n)\) is the current traffic and \(\epsilon\) is the modification step. \(F\) is either the gradient of a cost function or a heuristic rule. \(x\) can be defined by a probability density function \(p(x)\), and \(w\) the parameters of the learning system. The concept behind our system is based on the fact that each iteration uses an example drawn from the real flow instead of a finite training set. The average update therefore is a gradient descent algorithm which directly optimizes the expected risk.

For a given state of the system, we can define a local cost function \(J(x,w)\) which measures how well our system behaves on \(x\). The goal of learning is often the optimization of some functional of the parameters and of the concept to learn which we will call the global cost function. It is usually the expectation of the local cost function over the space \(X\) of the concept to learn:

\[
C(w) = \int_{X} J(x,w) p(x) dx
\]

Most often we do not know the explicit form of \(p(x)\) and therefore \(C(w)\) is unknown. Our knowledge of the random process comes from a series of observations \(\{x_i\}\) of the variable \(x\). We are thus only able to measure the realizations \(J(x,w)\) for the observations \(\{x_i\}\). A necessary condition of optimality for the parameters of the system is:

\[
\nabla C(w) = \mathbb{E}_X \{ \nabla J(x,w) \} = 0
\]

Where \(\nabla\) is the gradient operator. Because we do not know \(\nabla C(w)\) but only the realizations \(\nabla J(x,w)\), we cannot use classical optimization techniques to reach the optimum. One solution is to resort to adaptive algorithm, the simplest being the following:

\[
w(n) = w(n-1) - \gamma(n) \nabla J(x(n),w(n-1))
\]

where \(\gamma\) is the gradient step. There is an obvious similarity between (1) and (4) and when learning in the second stage aims at minimizing a cost function, there is a complete identity. The convergence proof of the online gradient descent of this formula can be found on [9]. This latter proofs that this formulation is particularly adequate for describing adaptive algorithms that simultaneously process an observation and learn to perform better. Such adaptive algorithms are very useful for tracking a phenomenon that evolves in time.

**AV-BW Delay Q-Routing** contains three modules distributed on the two stages of our functional model:

1. The objective of the first one is to select the K best candidate paths in terms of cumulative link cost (considered as static criterion) path from the source and the destination node (for simplicity, we consider here all link costs equal to 1 is done regarding the hop count principle).

2. The second stage contain two modules and consider the dynamic criteria: 1) It’s used in order to select N best paths among the first K best paths (N<K) according the estimation of residual bandwidth.

---

978-1-4244-5637-6/10/$26.00 ©2010 IEEE
A. The first stage: Constructing K Shortest Paths.

First of all, in spite of exploring the entire network environment which needs large computational time and space memory, our approach reduces this environment to K Best no loop paths in terms of cumulative cost links. Thus, each router maintains a link state database of the network topology map. We used a label setting algorithm based on the optimality principle and a generalization of Dijkstra's algorithm.

To find these K best paths, a variant of the Dijkstra's algorithm was proposed in [16], which does not stop when the destination is reached, but continues until the destination has been reached K times. The K best paths are applied to the intermediate nodes on the path from source node to destination node, establishing a list of multiple sub-paths from the source to intermediate nodes. The space complexity is $O(Knm)$, where $K$ is the number of paths, $n$ (resp. $m$) is the number of nodes (resp. the number of links). By using a pertinent data structure [17], and the principle of Pareto non-dominated paths [18] to reduce the search-space without compromising the solution, the time complexity can be kept at $O(Kn\log(Kn)+K^2m)$ [19].

When a network link changes its state (i.e., goes up or down, or its utilization is increased or decreased), the network is flooded with a link state advertisement (LSA) message. This message can be issued periodically or when the actual link state change exceeds a certain relative or absolute threshold. Obviously, there is tradeoff between the frequency of state updates (the accuracy of the link state database) and the cost of performing those updates. In our approach, the link state updates (the accuracy of the link state database) and the cost of state change exceeds a certain relative or absolute threshold.

When a network link changes its state, the router computes the $K$ optimal paths.

B. The second stage: Q learning algorithm to optimize the residual bandwidth and the end-to-end delay

1) First Module: Computing the end-to-end residual bandwidth

The objective of this module is to select a subset (N) of all candidate paths (K) issued by a first stage using a cost function based on end-to-end residual bandwidth. Several papers discuss the technique to estimate this parameter [20, 21]. Our solution is based on the reinforcement learning approach. Each router $x$ maintains a $BW$-table a collection of values of $BW_{x}(d,s)$ for every destination $d$ and for every interface $s$. This value reflects a residual bandwidth along a way to destination $d$ via interface $s$.

The router $x$ forwards the packet to the next best router $y$ with optimal metric (enough residual bandwidth). Just after receiving this packet, the router $y$ provides router $x$ with an estimate of its best $BW$-value to reach the destination (by sending reinforcement learning signal). This new information is then added to the $BW$-values of the router $x$.

Thus, the reinforcement signal $T_1$ is defined as the minimum between $BPR_{zd}$ (the estimated residual bandwidth along the way to the destination $d$, by the router $z$) and $BP_{z}$ (the residual bandwidth between nodes $y$ and $z$). This value is calculated as follows:

$$T_1(y, z, d) = \min (BP_{z}, BPR_{zd})$$  \hspace{1cm} (5)

Once the choice of the next router made, the router $s$ puts the packet in the waiting queue, and sends back the value $T_1$ as a reinforcement signal to the router $x$. It can therefore update its reinforcement function as:

$$BW_{x}(d,s) = \min (BP_{y}, T_1(y, z, d))$$ \hspace{1cm} (6)

The selection of the $N$ best paths is done based on $BW$.


The objective of this second module is to distribute the traffic on the $N$ best paths issued from the first stage. In order to take into account the dynamics of irregular traffic, and to force the router to take the alternative routes found in the second stage and not only the best path, we automatically compute a load balancing distribution on the obtained $N$ paths based on delay requirements. In this way packets reach their destinations with times close to optimal, while ensuring a good exploration of the remaining paths. The process is based on the packet delivery time (delay from the considered router to a destination node) computed by reinforcement learning and the latency in queue associated for each path in the considered router.

Let’s $D_i(t)$ be the packet delivery time for path $i$ at time $t$ computed above. Let $T'_{i}(t)$ be the latency in the queue associated with the closest router $n'$ in the direction of the path $i$ at time $t$ (that is, the neighbour of router $n$). The following formula allows us to compute the probability $P_{i}^{n}(t)$ for the $in$ path in router $n$ at time $t$:

$$P_{i}^{n}(t) = \frac{\beta}{\sum_{i=1}^{K} \left[1/D_i(t)\right]^\alpha \left[1/T'_{i}(t)\right]^\beta}$$ \hspace{1cm} (7)

$\alpha$ and $\beta$ are two tunable parameters that determine respectively the influence of delay time and waited queue time. They have an equivalent influence in the case of $\alpha = \beta$. This formula associates a very small probability for paths with high delay time and/or high queue time. This is due to the fact that when delay time (respectively waited time) increase the value of $\left[1/D_i(t)\right]^\alpha$ (respectively $\left[1/T'_{i}(t)\right]^\beta$) decreases.
IV. SIMULATION

The comparison with other well known algorithms developed in the literature was already presented in a past paper [6]. The results of the old version of KOQRA, with cumulative cost path and end-to-end delay parameters, were better than the so obtained with these algorithms. In this paper, to validate our results, we have focused only on the comparison between the new KOQRA system, called AV-BW Delay Q-Routing, based on three criteria (cumulative cost path, end-to-end delay, and residual bandwidth) and the oldest one (without the residual bandwidth), called KOQRA.

All algorithms have been implemented using OPNET software of MIL3 Company and we used the same data structure.

We tested the AV-BW Delay Q-Routing algorithm on a variety of network topologies. Performances of the algorithm are evaluated in terms of average packet delivery time and compared with KOQRA algorithms. In this goal, we used a random high interconnected network topology which includes 80 nodes pictured in Fig. 2. Two kinds of traffic have been studied: Continuous high load and Peak of high load of the network.

Fig. 3 and Fig.4 illustrates the average packet delivery time obtained when a congestion of the network is generated.

In the case of a low load, we note that KOQRA exhibits better performances than the algorithm based on AV-BW Delay Q-Routing. Thus, average packet delivery time obtained by AV-BW Delay Q-Routing is increased by 13% compared to the KOQRA routing policy. Indeed, it appears that under low traffic load, the variation of the residual bandwidth is negligible and the method integrating this variation requires more time.

In the case where the traffic network becomes very important and durable, the results illustrated in Fig.3 and Fig.4 show distinctly that the AV-BW Delay Q-Routing algorithm gives better performances in terms of average packet delivery time compared to the KOQRA policy. For example, after 1 hour of simulation, AV-BW Delay Q-Routing exhibits a performance of 60% higher than that of KOQRA. Indeed, the utilization of residual bandwidth of the way in the decision of routing, allows anticipation of router’s congestion.

The last test addresses the case of dynamic link failures. The tests compare the performances of our proposed approach (using the network topology presented in figure 5 which consists on two sides connected by 4 different links). The learning process begins with the network topology (top figure) shown in figure 5. During the simulation process, the network topology changes to a new topology (formed by eliminating one link between the two sides of the proposed network (link between router14 and router15)) shown at the bottom of figure 5.

Fig. 6 shows that the average packet delivery times increase when the link or the node is down. The results illustrated in this figure show also that the AV-BW Delay Q-Routing algorithm gives better performance in terms of average packet delivery time compared to the KOQRA policy when the phase of learning the solution of the link failure in order to establish alternative routes is finished. For example, after 30 minutes of simulation, AV-BW Delay Q-Routing delivers a 45% performance improvement over the past version of KOQRA.
Fig 5. Link’s failure topology

Fig 6. Performance for link failure scenario

V. CONCLUSION

In this paper, we present an evolution of our KOQRA routing algorithm based on a multi-path routing approach combined with the Q-Routing algorithm. Our approach is able to route efficiently even when critical aspects are allowed to vary dynamically in heterogeneous environment. The fact that the reinforcement signal is continuously updated, parameter adaptations of the proposed approach take into account dynamic variations of the network traffic.

Finally, extensions to the framework for using these techniques across hybrid networks to achieve end-to-end QoS needs to be investigated particularly for large-scale networks. Another challenging area concerns the conditioning of different models in order to take into account other parameters such as the information type of each packet (real-time, VBR, etc.) flow and routes the packet based on its information.

REFERENCES


