18

Inductive Approaches Based on Trial/Error Paradigm for Communications Network

Abdelhamid Mellouk
LISSI/SCTIC/QoSiDiN laboratory, University Paris XII-Val de Marne
France

1. Introduction

Today, providing a good quality of service (QoS) in irregular traffic networks is an important challenge. Besides, the impressive emergence and the important demand of the rising generation of real-time Multi-service (such as Data, Voice VoD, Video-Conference, etc.) over communication heterogeneous networks, require scalability while considering a continuous QoS. This emergence of rising generation Internet required intensive studies these last years which were based on QoS routing for heterogeneous networks on the one hand and on the backbone architecture level of communication networks characterized by a high and irregular traffic on the other hand (Mellouk et al., 2007b).

The basic function of QoS routing is to find a network path which satisfies the given constraints and optimize the resource utilization. The integration of QoS parameters increases the complexity of the used routing algorithms. Thus, the problem of determining a QoS route that satisfies two or more path constraints (for example, delay and cost) is known to be NP-complete (Gravey & Jhonson, 1979). A difficulty is that the time required to solve the Multi-Constrained Optimal path problem exactly cannot be upper-bounded by a polynomial function. Hence the focus has been on the development of pseudo-polynomial time algorithms, heuristics and approximation algorithms for multi-constrained QoS paths (Kuipers & Mieghem, 2005).

At present, several studies have been conducted on QoS routing algorithms which integrate the QoS requirements problematic for the routing algorithm. (Song & Sahni, 2006) introduce heuristics to find a source-to-destination path that satisfies two or more additive constraints on edge weights. (Jaffe, 1984) has proposed a polynomial time approximation algorithm for k multi-constrained path which uses a shortest path algorithm such as Dijkstra's (Sahni, 2005). (Korkmaz & Krunz, 2001) propose a randomized heuristic that employs two phases. In the first one, a shortest path is computed for each of the k QoS constraints as well as for a linear combination of all k constraints. The second phase performs a randomized breadth-first search for a solution of k multi-constrained problem. In (Kuipers & Mieghem, 2005), authors suggest that QoS routing in realistic networks could not be NP-complete regarding to a particular class of networks (topology and link weight structure).

Due this complexity, QoS routing problems are divided on several classes according to some aspects. For example, we distinguish the single path routing problem and the multipath routing problem, where routers maintain multiple distinct paths of arbitrary costs between a

Source: Reinforcement Learning: Theory and Applications, Book edited by Cornelius Weber, Mark Elshaw and Norbert Michael Mayer
ISBN 978-3-902613-14-1, pp.424, January 2008, I-Tech Education and Publishing, Vienna, Austria

source and a destination. The Multipath routing offers several advantages like good bandwidth, bounding delay variation, minimizing delay, and improved fault tolerance. So, it makes an effective use of the graph structure on a network, as opposed to single path routing which superimposes a logical routing tree upon the network topology. We find in literature many and various approaches that have been proposed to take into account the QoS requirement. The reader can refer to (Masip-Bruin et al., 2006) for an overview. Constraints imposed by QoS requirements, such as bandwidth, delay, or loss, are referred to as QoS constraints, and the associated routing is referred to as QoS routing which is a part of Constrained-Based Routing (CBR). Interest in constrained-based routing has been steadily growing in the Networks. Based on heuristics used in all of these approaches to reduce their complexity, we can classified it in three main categories:

Label Switching/Reservation Approaches- spurred by approaches like ATM PNNI, MPLS or GMPLS. With MPLS, fixed length labels are attached to packets at an ingress router, and forwarding decisions are based on these labels in the interior routers of the label-switched path. MPLS Traffic Engineering allows overriding the default routing protocol, thus forwarding over paths not normally considered. A resource reservation protocol such as RSVP must be employed to reserve the required resources. Another Architecture proposed for providing Internet QoS is the Differentiated Services architecture. Diffserv scales well by pushing complexity to network domain boundaries.

Multi-Constrained Path Approaches (MCP)- The goal of all of these approaches is to retrieve the shortest path among the set of feasible paths between two nodes. Considerable work in the literature has focused on a special case of the MCP problem known as the Restricted Shortest Path (RSP) problem. The goal is to find the least-cost path among those that satisfy only one constraint. An overview of these approaches can be found in (Kuipers et al., 2004).

Inductive approaches- To be able to make an optimal routing decision, according to relevant performance criteria, a network node requires to have a complete knowledge of the entire network state and an accurate prediction of the evolution of the networks and its dynamics. This, however, is impossible unless the routing algorithm is capable of adapting to the network state changes in almost real time. Thus, it is necessary to design intelligent and adaptive optimizing routing algorithms which take into account the network state and its evolution. We need to talk about QoS based state dependent routing algorithm.

In this chapter, we present an accurate description of the current state-of-the-art and give an overview of our work in the use of reinforcement learning concepts focused on communication networks. We focus our attention by developing a system based on this paradigm called KOCRA for K Optimal Constrained path Routing Algorithm. Basically, these inductive approaches selects routes based on flow QoS requirements and network resource availability. After developing in section 2 the concept of routing in high speed networks, we present in section 3 the family of inductive approaches. After, we present our works based on reinforcement learning approaches in three different communication networking domains: wired networks, mobile ad hoc networks, and packet router's scheduling networks. Last section concludes and gives some perspectives of this work.

2. Routing problem

As Internet is a large collection of more than 25,000 independent domains called autonomous systems (Ases), the cooperation between ASes is not optimized at the network

level, but rather it is based on the business relationships between organizations. The fully-independent management actions in each AS are expressed in terms of a policy-based routing strategy which primarily controls the outbound traffic of an AS and can include conflicting policies. A global solution for QoS routing over all the ASes must be able to handle both the differing QoS provisioning mechanisms and service specifications. This latter solution of building models of large ISP's is so complex to obtain (Quoitin & Uhlig, 2005). For this, Routing is divided onto two classes: IGP and EGP. IGP, such as OSPF or IS-IS, compute the interior paths in one AS, while EGP, such as BGP, is responsible for the selection of the interdomain paths. To fulfill application QoS requirements, many ISPs have deployed mechanisms to provide differentiated services in their networks. In fact, in the last decade, the development of none of QoS routing proposals has turned out to be sufficiently appealing to become deployed in practice. This is because ISPs have preferred to overprovision their networks rather than deliver and manage QoS (Yanuzzi et al., 2005).
In the IGP or EGP cases, a routing algorithm is based on the hop-by-hop shortest-path paradigm. The source of a packet specifies the address of the destination, and each router along the route forwards the packet to a neighbour located "closest" to the destination. The best optimal path is choosed according to given criteria. When the network is heavily loaded, some of the routers introduce an excessive delay while others are under-utilized. In some cases, this non-optimized usage of the network resources may introduce not only excessive delays but also high packet loss rate. Among routing algorithms extensively employed in the same AS routers, one can note: distance vector algorithm such as RIP and the link state algorithm such as OSPF or IS-IS (Grover, 2003).

2.1 Distance vector approach

Also known as Bellman-Ford or Ford-Fulkerson, the heart of this type of algorithm is the routing table maintained by each host. With the distance-vector (DV) routing scheme (e.g. RIP, IGRP), each node exchanges with its neighbouring nodes its distance (e.g. hop count) to other networks. The neighbouring nodes use this information to determine their distance to theses networks. Subsequently these nodes share this information with their neighbours, etc. In this way the reachability information is disseminated through the networks. Eventually each node learns, which neighbour (i.e. next hop router) to use, to reach a particular destination with a minimum number of hops. A node does not learn about the intermediate to the destination. These approaches suffers from a classic convergence problem called "count to infinity". It also does not have an explicit information collection phase (it builds its routing table incrementally). DV routing protocols are designed to run on small networks.

2.2 Link state approach

With link-state (LS) routing (e.g. OSPF or IS-IS), each node builds a complete topology database of the network. This topology database is used to calculate the shortest path with Dijkstra's algorithm. Each node in the network transmits its connectivity information to each other node in the network. This type of exchange is referred to as flooding. This way each node is able to build a complete topological map of the network. The computational complexity cost used here is lower than the DV protocol. However, LS algorithms trade off communication bandwidth against computational time.

3. Inductive approaches

Modern communication networks is becoming a large complex distributed system composed by higher interoperating complex sub-systems based on several dynamic parameters. The drivers of this growth have included changes in technology and changes in regulation. In this context, the famous methodology approach that allows us to formulate this problem is dynamic programming which, however, is very complex to be solved exactly. The most popular formulation of the optimal distributed routing problem in a data network is based on a multicommodity flow optimization whereby a separable objective function is minimized with respect to the types of flow subject to multicommodity flow constraints (Gallager, 1977; Ozdaglar & Bertsekas, 2003). In order to design adaptive algorithms for dynamic networks routing problems, many of works are largely oriented and based on the Reinforcement Learning (RL) notion (Sutton & Barto, 1997). The salient feature of RL algorithms is the nature of their routing table entries which are probabilistic. In such algorithms, to improve the routing decision quality, a router tries out different links to see if they produce good routes. This mode of operation is called exploration. Information learnt during this exploration phase is used to take future decisions. This mode of operation is called exploitation. Both exploration and exploitation phases are necessary for effective routing and the choice of the outgoing interface is the action taken by the router. In RL algorithms, those learning and evaluation modes are assumed to happen continually. Note that, the RL algorithms assigns credit to actions based on reinforcement from the environment. In the case where such credit assignment is conducted systematically over large number of routing decisions, so that all actions have been sufficiently explored, RL algorithms converge to solve stochastic shortest path routing problems. Finally, algorithms for RL are distributed algorithms that take into account the dynamics of the network where initially no model of the network dynamics is assumed to be given. Then, the RL algorithm has to sample, estimate and build the model of pertinent aspects of the environment.

Many of works has done to investigate the use of inductive 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 that they request, and to improve network provide robustness and resilience. For example, we can note the following approaches:

Q-Routing approach- In this technique (Boyan & Littman, 1994), each node makes its routing decision based on the local routing information, represented as a table of Q values which estimate the quality of the alternative routes. These values are updated each time the node sends a packet to one of its neighbors. However, when a Q value is not updated for a long time, it does not necessarily reflect the current state of the network and hence a routing decision based on such an unreliable Q value will not be accurate. The update rule in Q-Routing does not take into account the reliability of the estimated or updated Q value because it depends on the traffic pattern, and load levels. In fact, most of the Q values in the network are unreliable. For this purpose, other algorithms have been proposed like Confidence based Q-Routing (CQ-Routing) or Confidence based Dual Reinforcement Q-Routing (DRQ-Routing).

Cognitive Packet Networks (CPN)- CPNs (Gelenbe et al., 2002) are based on random neural networks. These are store-and-forward packet networks in which intelligence is constructed into the packets, rather than at the routers or in the high-level protocols. CPN is then a reliable packet network infrastructure, which incorporates packet loss and delays directly

into user QoS criteria and use these criteria to conduct routing. Cognitive packet networks carry three major types of packets: smart packets, dumb packets and acknowledgments (ACK). Smart or cognitive packets route themselves, they learn to avoid link and node failures and congestion and to avoid being lost. They learn from their own observations about the network and/or from the experience of other packets. They rely minimally on routers. The major drawback of algorithms based on cognitive packet networks is the convergence time, which is very important when the network is heavily loaded.

Swarm Ant Colony Optimization (AntNet)- Ants routing algorithms (Dorigo & Stüzle, 2004) are inspired by dynamics of how ant colonies learn the shortest route to food source using very little state and computation. Instead of having fixed next-hop value, the routing table will have multiple next-hop choices for a destination, with each candidate associated with a possibility, which indicates the goodness of choosing this hop as the next hop in favor to form the shortest path. Given a specified source node and destination node, the source node will send out some kind of ant packets based on the possibility entries on its own routing table. Those ants will explore the routes in the network. They can memory the hops they have passed. When an ant packet reaches the destination node, the ant packet will return to the source node along the same route. Along the way back to the destination node, the ant packet will change the routing table for every node it passes by. The rules of updating the routing tables are: increase the possibility of the hop it comes from while decrease the possibilities of other candidates. Ants approach is immune to the sub-optimal route problem since it explores, at all times, all paths of the network. Although, the traffic generated by ant algorithms is more important than the traffic of the concurrent approaches.

In the following, we give an overview of our work in the use of reinforcement learning concepts focused on communication networks. We focus our attention by developing a system based on this paradigm called KOCRA for K Optimal Constrained path Routing Algorithm and present our works based on reinforcement learning approaches in three different communication networking domains: wired networks, mobile ad hoc networks, and packet router's scheduling networks.

4. KOCRA system based reinforcement learning in routing wired networks.

KOCRA is the successor of KONRS, a K Optimal Neural Routing System (Mellouk et al., 2006a).

4.1 Brief summary of KONRS

In (Mellouk et al., 2006a), we have presented an adaptive routing algorithm based on Q learning approach, the Q function is approximated by a reinforcement learning based neural network (NN). As shown in figure 1, In this approach, NN ensure the prediction of parameters depending on traffic variations. Compared to the approaches based on a Q table, the Q value is approximated by a reinforcement learning based neural network of a fixed size, allowing the learner to incorporate various parameters such as local queue size and time of day, into its distance estimation. Indeed, a neural network allows the modelling of complex functions with a good precision along with a discriminating training and a taking into account of the context of the network. Moreover, it can be used to predict non-stationary or irregular traffics. In this approach, the objective is to minimize the average packet delivery time. Consequently, the reinforcement signal which is chosen corresponds

to the estimated time to transfer a packet to its destination. Typically, the packet delivery time includes three variables: The packet transmission time, the packet treatment time in the router and the latency in the waiting queue.

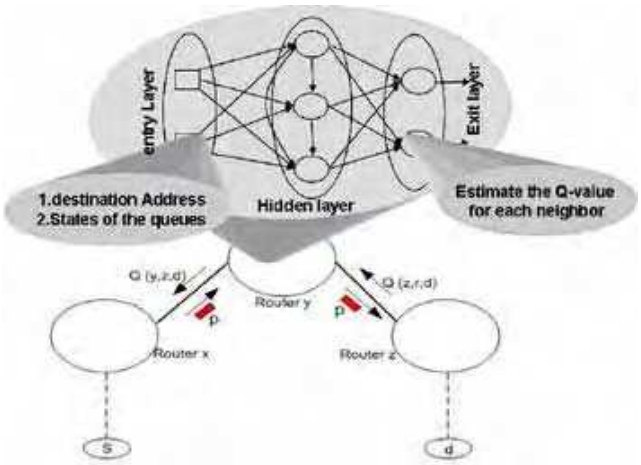

Fig. 1. Neural Net Architecture

The input cells in NN used correspond to the destination and the waiting queue states. The outputs are the estimated packet transfer times passing through the neighbours of the considered router. The algorithm derived from this architecture can be described according to the following steps:

When receiving a packet of information:
1. Extract a destination IP address,
2. Calculate Neural Network outputs,
3. Select the smallest output value and get an IP address of the associated router,
4. Send the packet to this router,
5. Get an IP address of the precedent router,
6. Create and send the packet as a reinforcement signal.

At the reception of a reinforcement signal packet:
1. Extract a Q estimated value computed by the neighbor,
2. Extract a destination IP address,
3. Neural Network updating using a retro-propagation algorithm based on gradient method,
4. Destroy the reinforcement packet.

This approach offers advantages compared to standard Distance Vector (DV) routing policy and earlier Q-routing algorithm, like the reduction of the memory space for the storage of secondary paths, and a reasonable computing time for alternative paths research. The Q value is approximated by a reinforcement learning based neural network of a fixed size. Results given in (Mellouk et al., 2006a) show better performances of the proposed algorithm

comparatively to standard distance vector and Q-routing algorithms. In fact, at a high load level, the traffic is better distributed along the possible paths, avoiding the congestion of the network.

4.2 The concepts behind KOCRA

This first version of our KONRS routing system explore all the network environment and do not take into account loop problem in a way leading to large time of convergence algorithm. To address this drawback and reducing computational time, we have worked on the evolution of our earlier Q-neural routing algorithm and present the enhanced version of KONRS called "K Optimal Constrained path Routing Algorithm (KOCRA)" (Mellouk et al., 2007a). KOCRA contains three stages. The objective of the first stage is to select the K Best candidate paths according to the cost cumulative path from the source and the destination nodes (for simplicity, we consider here all link costs equal to 1). The second stage is used to integrate the dynamics of traffic. For this, a continuous end-to-end delay among the K Best selected Paths is computed using a reinforcement Q-learning function. In order to force the router to take the alternative routes regarding to the second stage, we used a third one which compute automatically a probability affected to each path based on packet delivery time obtained by the second stage and the time latency in queuing file associated for each path.

4.2.1 First stage: constructing K-best paths

First of all, in spite of exploring the entire network environment which needs large computational time and space memory (Mellouk et al., 2006a), our approach reduces this environment to K Best no loop paths in terms of cost cumulative links. Thus, each router maintains a link state database as map of the network topology. We used a label setting algorithm based on the optimality principle and being a generalization of Dijkstra's algorithm (Sahni, 2005). In order to find these K best paths, a variant of Dijkstra's algorithm proposed in (Eppstein, 1999) was used. The space complexity is $O(Kmn)$, where K is the number of paths, m (resp. n) is the number of edges (resp. the number of links). By using a pertinent data structure, the time complexity can be kept at $O(m+nlogn+K)$ (Mellouk et al., 2007a). 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 information is updated when the actual link state change. Once the link state database at each router is updated, the router computes the K optimal paths.

Let a DAG (N; A) denote a network with n nodes and m edges, where N = {1.. n}, and A ={aij/i,j€N}. N}. The problem is to find the top K paths from source s to all the other nodes. Let's define a label set X and a one-to-many projection h: N→ X, meaning that each node i€ N corresponds to a set of labels h(i), each element of which represents a path from s to i.

---

* *S the source node*
* *N –set of nodes in network*

```
* X – the label set
* Count_i – Number of paths determined from S to I
* elm – Affected number to assigned label
* P – Paths list from S to destination (D)
* K – paths number to compute
      * h – corresponding between node and affected label number
/* Initialisation */
count_i = 0 /* for all i □ K */
elem = 1
h(elem) = s
h^-1(s) = {elem}
distance_elem = 0
X = {elem}
P^K = 0
While (count_t < K and X != { })
    begin
        /* find a label lb from X, such that
        distance_lb <= distance_lb1, □ lb1 □ X*/
        X = X – {lb}
        i = h(lb)
        count_i = count_i + 1
        if (i == D) then /* if the node I is the destination node D */
                begin
                        p = path for 1 to lb
                        P^K = P^K U {h(p)}
                end
        if (count_i <= K) then
              begin
                for each arc(i,j) □ A
                      begin
                       /* Verify if new label does not result in loop */
                       v=lb
                       While (h(v) != s)
                        begin
                          if (h(v) == j) then
                             begin
                                goto do_not_add
                             end
                           v = previous_v
                        end
                    /* Save information from new label */
                    elem = elem + 1
                    distance_elem = distance_n + c_ij
                    previous_elem = lb
                    h(elem) = j
                    h^-1(j) = h^-1(j) U {elem}
```

```
            X = X U {elem}
            do_not_add:
                        end
end
end
```

4.2.2 Second stage: Q-learning algorithm for optimizing the end-to-end delay

After finding our K best Optimal Paths based on link costs, the second step is to distribute the traffic on these K candidate paths. For this, we use another criteria based on the end-to-end delay. The reinforcement signal which is chosen corresponds to the estimated time to transfer a packet to its destination. This value is computed by a variant of Q-Routing algorithm which is considered as an asynchronous relaxation of the Bellman-Ford algorithm used in distance vector protocols. Typically, the packet delivery time includes three variables: the packet transmission time, the packet treatment time in the router and the latency in the waiting queue. In our case, the packet transmission time is not taken into account. In fact, this parameter can be neglected in comparison to the other ones and has no effect on the routing process.

In this approach, each router x maintains in a Q-table a collection of values of $Q(x, y, d)$ for every destination d and for every interface y. This value reflects a delay of delivering a packet for destination d via interface s. Then, the router x forwards the packet to the best next router y determined from the Q-table. Just after receiving this packet, the router y provides x an estimate of its best Q value to reach the destination. This new information is then added in the Q-values of the router x.

The reinforcement signal T employed in the Q-learning algorithm can be defined as the minimum of the sum of the estimated $Q(x, y, d)$ sent by the router y neighbour of router x and the latency in waiting queue q_x corresponding to router x.

$$T = \min_{y \in \text{neighboof x}} \{ \; q_x + Q(x,y,d) \; \} \tag{1}$$

Where $Q(x, y, d)$, denote the estimated time by the router x so that the packet p reaches its destination d through the router y. This parameter does not include the latency in the waiting queue of the router x. The packet is sent to the router y which determines the optimal path to send this packet.

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

$$\Delta Q(x,y,d) = \eta(\alpha + T - Q(x,y,d) \;) \tag{2}$$

α and η are the packet transmission time between x and y and the learning rate respectively. So, the new estimation $Q'(x,y,d)$ can be written as follows:

$$Q'(x,y,d) = Q(x,y,d)(1-\eta) + \eta(T+\alpha) \tag{3}$$

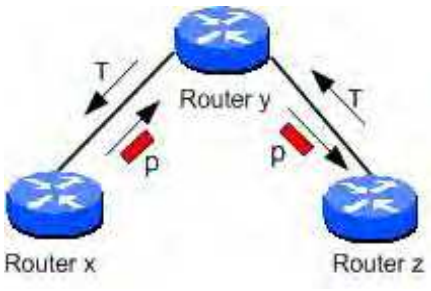

Fig. 2. Updating of the reinforcement signal.

4.2.3 Third stage: adaptive probabilistic path selection.

The goal of this stage is to distribute the traffic on K best paths in probabilistic manner. To force the router to take alternative routes find in K best paths and not only the best one, we compute a probability affected to each path automatically. We associated a maximal value P_{max} for the best path and divided the rest of probability (1- P_{max}) for the remaining (K-1) paths. The value of P_{max} is fixed by a counting process. To force the router to take the alternative routes find in the second stage and not only the best path, a uniform distributed random process is implemented in each router. This process chooses randomly a number between [0, 1]. Next, a router choose the path verifying the condition that it's probability is less than this random number. For example, in the situation characterized by K=2 (two paths), P1=0.8, P2=0.2, if the random number <=0.8, the router chooses the first path, otherwise the router takes the second one. In this manner, the flow packets reach their destination with a time close to optimal, while ensuring a good exploration of the remaining paths. Unfortunately, this kind of fixed hand probability don't take automatically into account the dynamic of the irregular traffic. We have proposed a second version of computing automatically the load balancing distribution. The process is based on the packet delivery time computed by our Q reinforcement learning and the latency in queuing file associated for each path.

Let $D_i(t)$ be the packet delivery time for path i at time t. Let $T_i^n(t)$ be the latency in queuing file associated to closest router n' in the direction of path i at time t (that is, the neighbour of router n). The following formula allows us to count the probability $P_i^n(t)$ for the i^{th} path in router n at time t:

$$P_i^n = \left[\left(\frac{1}{D_i}\right)^{\alpha} + \left(\frac{1}{T_i^n}\right)^{\beta}\right] \Big/ \left[\sum_{i=1}^{z}\left(\frac{1}{D_i}\right)^{\alpha} + \left(\frac{1}{T_i^n}\right)^{\beta}\right] \quad (4)$$

Where α and β are two tuneable parameters that determine respectively the influence of delay time and waited queue time. They have an equivalent influence in the case of $a=\beta$. 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 $[1/D_i(t)]^{\alpha}$ (respectively $[1/T_i(t)]^{\beta}$) decreases.

4.3 Performance evaluation

To validate our results in the case of irregular traffic in wired networks, we take the results given by a well-known Djikstra's algorithm (which offers to use an existing polynomial-time path computation) used in protocols such OSPF, IS-IS or CISCO EIGRP as a reference for our study. This choice of this classical approach is argued by the fact that the majority of ISP's used actually this kind of protocols to exchange routing information in their networks. In order to do comparison with KOCRA, parameters of standard approach used here are fixed in order to optimize the delay and cost criteria simultaneously (on the rest of paper, we used the notation "Standard Optimal Multi-Path Routing Algorithm (SOMRA)" for this kind of algorithm). All algorithms have been implemented with OPNET and used the same data structure. OPNET software constitutes for telecommunications networks an appropriate modelling, scheduling and simulation tool. It allows the visualization of a physical topology of a local, metropolitan, distant or on board network. The protocol specification language is based on a formal description of a finite state automaton.
The simulations presented in this article consisted of creating a traffic merged in irregular network topology, through which the two family of algorithms (KOCRA and SOMRA) computed the best paths between two nodes. QoS measures of each of tested algorithms concerns two additive constraints: cost and delay criteria. Results given in all the figures are evaluated in terms of average packet end-to-end delivery time on both topologies. Time simulation is represented on the other axis of the figures.

4.3.1 Simulation parameters on the irregular topology.

The topology of the network is specified by a collection of routers and a set of links that bind these routers elements. The network traffic is specified in the source router by setting several parameters like: the start time, the stop time, the statistical distribution for packet inter-arrival times, the statistical distribution for packet size and the destination node.
To ensure a meaningful validation of our algorithm performance, we devised a realistic simulation environment in terms of network characteristics, communications protocols and traffic patterns. We focus on IP datagram networks with irregular topology. The topology of the network employed for simulations includes 36 interconnected nodes with essentially two parts of the network, as shown in Fig 3. This topology is the same used in (Boyan & Littman, 1994) for their Q learning approach.
We model traffic in terms of requests characterized by its source and destination. While we concern ourselves with arrival and departure of flows, we do not model the data traffic of the flows. For simplicity, we also chose not to implement a proper management of error, flow and congestion control. In act, each additional control component has a considerable impact on the network performance, making very difficult to evaluate and to study properties of each control algorithm without taking in consideration the complex way it interacts with all the other control components (Dorigo & Stüzle, 2004). Therefore, we chose to test the behavior of our algorithm such that the routing component can be evaluated in isolation.
The traffic is sent/received by four end nodes (marked in the figure noeud100, noeud101, noeud102 and noeud103).
For our simulation results, we studied the performance of the algorithms for increasing traffic load, examining the evolution of the network status toward a saturation condition,

and for temporary saturation conditions. For this topology, we study the performance of our routing strategies according a Poisson Law inter-arrival times statistical distribution.

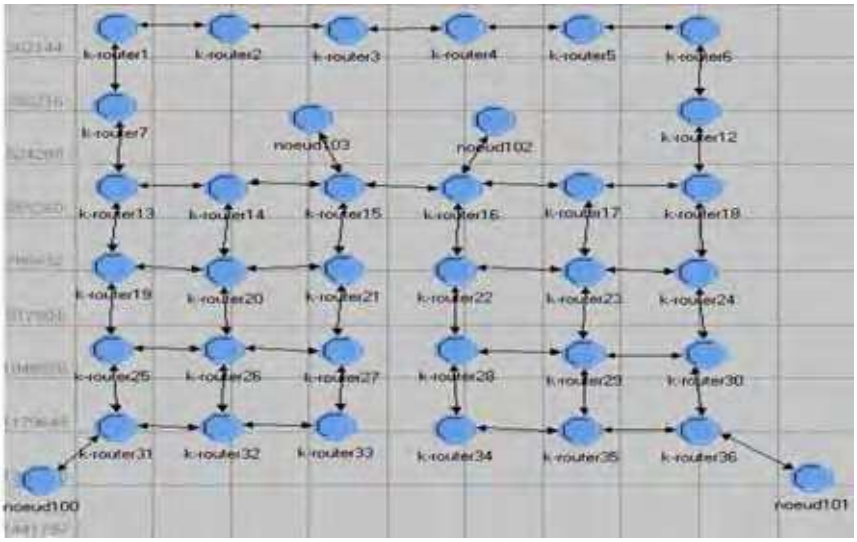

Fig. 3. Network topology.

4.3.2 Poisson distribution of model traffic.

In probability theory, the Poisson distribution is a discrete probability distribution which expresses the probability of a number of events occurring in a fixed period of time if these events occur with a known average rate, and are independent of the time since the last event. The It is represented by random variables N that count a number of discrete occurrences (called "arrivals") that take place during a time-interval of given length. The probability that there are exactly k occurrences (with k a non-negative integer, k = 0, 1, 2, ...) is:

$$P(k,\lambda) = \frac{e^{-\lambda}\lambda^{k}}{k!} \tag{5}$$

Where λ is a positive real number and is the mean number of occurrences k. The Poisson law is then defined by its mean λ parameter.

In our simulations, we suppose the mean λ of the inter-arrival times is 3 s and fix the time start to 1 min and the stop time to the end of simulation is fixed to 6 h.

4.3.3 Simulation results.

As shown in Fig. 4 which represent time simulation versus the average packet delivery time, our probabilistic K Optimal Constrained path Routing Algorithm (KOCRA) give better results than the well-known N best optimal path routing Algorithm SOMRA. This is due to the fact that in our new approach, routers are able to take into account not only the average of delivery delay but also the waiting queue time. Thus, they are able to adapt their

decisions very fast and in close concordance with the network dynamics. In spite of the many packages taking secondary ways, N-optimal routing does not present better performances because it rests on a probabilistic method to distribute the load of the network over the closest cost paths, and not on the degradation of the times of routing. So, in classical approach, the routers take their decisions only according to the average of delivery delay and the exploration of potentials good paths, none trivially best and that can give us betters results, is not realized. Our approach, with the introduction of a probabilistic module, responds to this inconvenience and shows better results for Poisson law distribution of traffic. Thus, mean of average packet delivery time obtained by KOCRA is reduced by 37% compared to traditional N best optimal routing Algorithm.

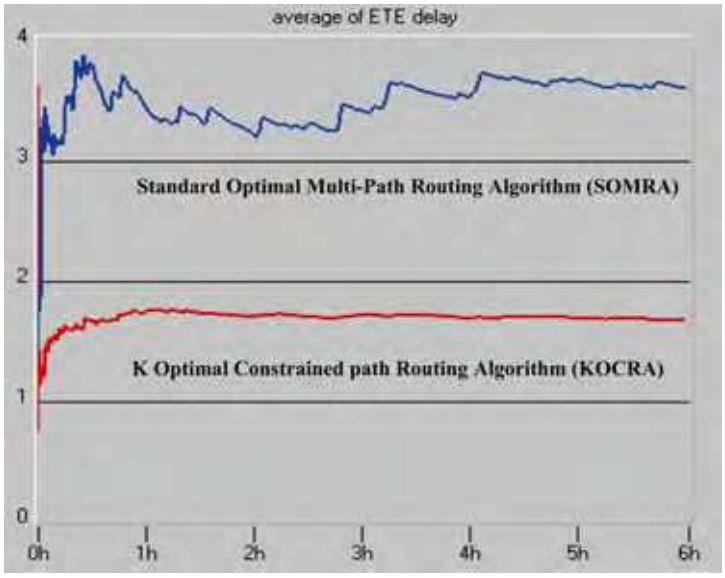

Fig. 4. Poisson law distribution simulations results

5. AMDR based reinforcement learning in mobile ad hoc networks.

AMDR (Adaptive Mean Delay Routing) is a new adaptive routing protocol based on probabilities and built around two exploration RL agents. Exploration agents gather mean delay information available at each node in their route and calculate total delay between source and destination. According to the delay value gathered, probabilistic routing tables are updated at each intermediate node. In order to deal with mobile nodes synchronisation we consider, in our protocol, delay estimation model proposed in [Naimi, 2005; Naimi & Jacquet, 2004], instead of instantaneous delay considered in the most oriented delay routing protocols.

Unlike data packets, control packets, used in adaptive routing, are sent in broadcast manner and so treated at IEEE 802.11, MAC layer differently than unicast packets. For this, we consider that trip delay of a control packet is not the same of a data packet.

In AMDR, routing function is determined by means of very complex interactions of forward and backward network exploration agents. Forward agents report network delay conditions to the backward ones. So, no node routing updates are performed by the forward agents.
AMDR uses two kinds of agents: Forward Exploration Packets (FEP) and Backward Exploration Packets (BEP). Forward agents explore the paths of the network, for the first time in reactive manner, but it continues the exploration proactively.
FEP packets create a probability distribution at each node for its neighbours. Backward agents are used to propagate the information gathered by forward agents through the network, and to adjust the routing table entries.

5.1 Forward exploration packet

When a new traffic arrives at a node *n*, periodically the node *n* generates a forward exploration packet called FEP. The FEP packet is then sent to the destination of traffic. Each FEP packet contains the following fields: *source_node_address, destination_node_address, ext_hop_address, stack_of_visiting_nodes_addresses, total_delay*.
If the entry of the current destination does not exist then a routing table entry is immediately created. The algorithm of FEP sending is the following:

```
Algorithm (Send FEP)
    At Each T_interval_secondes Do
    Begin
        Generate a FEP
        If any entry for this destination Then
                Create an entry with uniform probabilities.
      End If
        Broadcast the FEP
    End
End (Send FEP)
```

When a FEP arrives to a node *i*, it checks if the address of the node *i* is not equal to the *destination address* contained in the FEP agent then the FEP packet will be forwarded. FEP packets are forwarded according to the following algorithm:

```
Algorithm (Receive FEP)
        If any entry for this destination Then
                Create an entry with uniform probabilities.
        ELSE If my_adress ≠ dest_adress Then
            IF FEP not already received then
                    Stock address of the current node,
                    Recover the available mean delay,
                    Broadcast FEP
            ELSE
                    Send BEP
          End IF
      End IF
End (forward FEP)
```

5.2 Backward exploration packet

As soon as a forward agent FEP reaches its destination, a backward agent called BEP is generated and the forward agent FEP is destroyed. BEP inherits the stack and the total delay information contained in the forward agent. We define five options for our algorithm in order to reply to a FEP agent. The algorithm of sending a BEP packet depends on the chosen option. The five options considered in our protocol are:

Reply to All: for each FEP reception, the destination a BEP packet is generated. In this case, the delay information is not used and the overhead generated is very important.

Reply to First: Only one BEP agent is generated for a FEP packet. The mean delay module is not exploited. It's the same approach used in the AntNet. The overhead is reduced but any guarantee to have the best delay paths.

Reply to N: destination node can generate until N BEP packet for the same FEP. The overhead is reduced compared to reply to N but it is more important than the Reply to first option.

Reply to the Best: We save at each node the information of the best delay called "Node.Total_delay". When the first FEP arrives to the destination, Node.Total_delay takes the value of total delay of the FEP packet. When another FEP arrives, its total delay is compared to the node total delay, and we reply only if the FEP has a delay better or equal to the node total delay.

Reply to delay Constraint: This option focuses on Delay-Constrained-Path (DCP) unicast routing. The DCP issue is to select the path with given delay requirement. This is the case of real time applications having serious delay constraints.

Applications needs in term of delay are determined in a max delay supported "D" value. Arriving to destination, this one compares FEP *total_delay* to the application *delay_constraint* "D". If the FEP *total_delay* is equal or less than "D" then a BEP is generated and sent to the source of the FEP. We give in the following a part of BEP sending algorithm. We focus on "Reply to best" option which will be used in simulation part:

Algorithm (Send BEP)

```
Select Case Option
   Case: Reply to All
         ....
   Case: Reply to first
   If (First (FEP) ) Then
         …..
     endIf
   Case: Reply to N
   If (N>0) Then
         .......
     endIf
   Case: Reply to Best
   If (FEP.Total_Delay <= Node.Total_Delay) Then
         Genarate BEP
         BEP.Total_Delay=0,
         BEP.dest = FEP.src,
```

```
        Send BEP,
        Node.Total_Delay= FEP.Total_Delay,
    endIf
  Case: Reply to Delay Constraint (D)
  If (FEP.Total_Delay <= D) Then
        .....
    End If
End (Send BEP)
```

Backward Exploration Packet (BEP) retraces the inverse path traversed by the FEP packet. In other words, unlike FEP packets, a BEP packet is sent in a unicast manner because it must take the same path of its FEP generator. During its trip, the BEP agent calculates the total mean delay of its route and uses this new delay to adjust the probabilistic routing table of each intermediate node. The algorithm of forwarding BEP agent is the following:

```
Algorithm (Receive BEP)
    If (my_adress = BEP.dest) Then
        Update probabilistic routing table
    Else
        Update probabilistic routing table
        Forward BEP
    End If
End (Receive BEP)
```

5.3 Updating routing tables

Routing tables are updated when a BEP agent is received. The probabilities updating can take many forms, and we have chosen updating rules (6), (7), (8) and (9) described in (Baras & Mehta, 2003). As soon as, routing table is calculated, data packets are then routed according to the highest probabilities in the probabilistic routing tables.

Unlike on demand routing protocols, there is no guarantee to route all packets on the same route because of the proactive exploration. The BEP agent make changes to the probability values at the intermediate and final node according to the following update rules:

$$p_{fd} \leftarrow (p_{fd} + r)\,(1+r) \tag{6}$$

$$p_{nd} \leftarrow p_{nd}/(1+r) \tag{7}$$

$$p_{nd} \leftarrow p_{nd} - rp_{nd} \tag{8}$$

$$p_{fd} \leftarrow p_{fd} + r\,(1-p_{fd}) \tag{9}$$

In both the above cases, the reinforcement parameter *r* can be defined as a function of delay. Here, r=k /f(c), where k > 0 and f(c) is the cost function (Baras & Mehta, 2003).

5.4 Flooding optimization

In order to improve the performance of our routing protocol, we introduce the MPR (Nguyen & Minet, 2006) concept in the broadcast process. However, the MPR selection

according to native OLSR is unable to build path satisfying a given QoS request. To avoid this problem, we propose a new algorithm for MPR selection. We keep at each node a table called MPR table containing a partial view of MPR neighbours. Our algorithm takes into account the mean delay available at each node. The MPR selection algorithm based on mean delay is the same proposed for bandwidth in (Nguyen & Minet, 2006), unlike their approach for bandwidth MPR; we define only one kind of MPR which are delay MPR. Mean delay MPR selection algorithm is composed of the following steps:

1. A node N_i selects, first, all its neighbours that are the only neighbours of a two hop node from N_i.
2. Sort the remaining one-hop delay neighbours in increasing order of mean delay.
3. Consider each one-hop neighbour in that order: this neighbour is selected as MPR if it covers at least one two-hop neighbour that has not yet been covered by the previous MPR.
4. Mark all the selected node neighbours as covered and repeat step 3 until all two-hop neighbours are covered.

With the present MPR selection algorithm, we guarantee that paths having best delays will be discovered but there are any guarantees about the overhead generated (Ziane & Mellouk, 2006).

5.5 Performance evaluation

We use NS-2 simulator to implement and test AMDR protocol. We present in this section two scenarios of simulation. In the first one, we define a static topology of 8 nodes. To compare AODV, OLSR and AMDR, we have chosen the Reply to Best option of AMDR.

5.5.1 Static scenario

The following table summarizes the simulation environment:

Routing	AODV, AMDR, OLSR
MAC Layer	802.11
Bandwidth	11Mb/s
TERRAIN	1000m,1000m
Nodes	8
Simulation time	1000 sec
Data traffic	exponential

Table 1. Simulation settings scenario 1

We injected three types of traffic in the network and compared for each simulation the file trace for each routing protocol. Figure 5 shows the comparison of end to end delay realized by AODV, AMDR and OLSR protocols. We can see that, at first OLSR realizes the best