1

Wireless Networks Inductive Routing Based on Reinforcement Learning Paradigms

Abdelhamid Mellouk
Network &Telecom Dept and LiSSi Laboratory
University Paris-Est Creteil (UPEC), IUT Creteil/Vitry,
France

1. Introduction

Wireless networks, especially Sensor ones (WSN), are a promising technology to monitor and collect specific measures in any environment. Several applications have already been envisioned, in a wide range of areas such as military, commercial, emergency, biology and health care applications. A sensor is a physical component able to accomplish three tasks: identify a physical quantity, treat any such information, and transmit this information to a sink (Kumar et al., 2008; Buford et al., 2009; Olfati-Saber et al., 2007).

Needs for QoS to guarantee the quality of real time services must take into account not only the static network parameters but also the dynamic ones. Therefore, QoS measures should be introduced to the network so that quality of real time services can be guaranteed. The most popular formulation of the optimal distributed routing problem in a data network is based on a multicommodity flow optimization whereby a separate objective function is minimized with respect to the types of flow subjected to multicommodity flow constraints. Given the complexity of this problem, due to the diversity of the QoS constraints, we focus our attention in this paper on bio-inspired QoS routing policies based on the Reinforcement Learning paradigm applied to Wireless Sensor Networks.

Many research works focus on the optimization of the energy consumption in sensor networks, as it directly affects the network lifetime. Routing protocols were proposed to minimize energy consumption while providing the necessary coverage and connectivity for the nodes to send data to the sink. Other routing protocols have also been proposed in WSN to improve other QoS constraints such as delay.

The problem here is that the complexity of routing protocols increases dramatically with the integration of more than one QoS parameter. Indeed, determining a QoS route that satisfies two or more **non-correlated constraints (for example, delay and bandwidth)** is an NP-complete problem (Mellouk et al., 2007), because the Multi-Constrained Optimal path problem cannot be solved in polynomial time. Therefore, research focus has shifted to the development of pseudopolynomial time algorithms, heuristics, and approximation algorithms for multi-constrained QoS paths.

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 Wireless Sensor Networks. We focus our attention by developing systems based on this paradigm called AMDR and EDAR. 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 wireless sensor networks, we present in section 3 the family of inductive approaches. After, we present in two sections our works based on reinforcement learning approaches. Last section concludes and gives some perspectives of this work.

2. Routing problem in Wireless Sensor Networks

Goal aspects that are identified as more suitable to optimize in WSNs are QoS metrics. Sensor nodes essentially move small amounts of data (bits) from one place to another. Therefore, equilibrium should be defined in QoS and energy consumption, to obtain meaningful information of data transmitted. Energy efficiency is an evident optimization metric in WSNs. More generally, several routing protocols in WSNs are influenced by several factors:

Minimum life of the system: In some cases, no human intervention is possible. Batteries of sensors can not be changed; the lifetime of the network must be maximized.

Fault tolerance: Sensor network should be tolerant to nodes failures so as the network routes information through other nodes.

Delay: Delay metric must be taken into account for real-time applications to ensure that data arrives on time.

Scalability: Routing protocols must be extendable, even with several thousand nodes.

Coverage: In WSN, each sensor node obtains a certain view of the environment. This latter is limited both in range and in accuracy (fig. 1); it can only cover a limited physical area of the environment. Hence, area coverage is also an important design parameter in WSNs. Each node receives a local view of its environment, limited by its scope and accuracy. The coverage of a large area is composed by the union of several smaller coverages.

Connectivity: Most WSNs have high density of sensors, thus precluding isolation of nodes. However, deployment, mobility and failures vary the topology of the network, so connectivity is not always assured (Tran et al., 2008).

Quality of Service: In some applications, data should be delivered within certain period of time from the moment it is sensed; otherwise the data will be useless. Therefore bounded latency for data delivery is another condition for time-constrained applications. However, in many applications, energy saving -which is directly related to the network's lifetime- is considered relatively more important than the quality of the transmitted data. As the energy gets depleted, the network may be required to reduce the quality of the results in order to reduce energy dissipation in nodes and hence lengthen network lifetime. Hence, energy-aware routing protocols are required to capture this requirement.

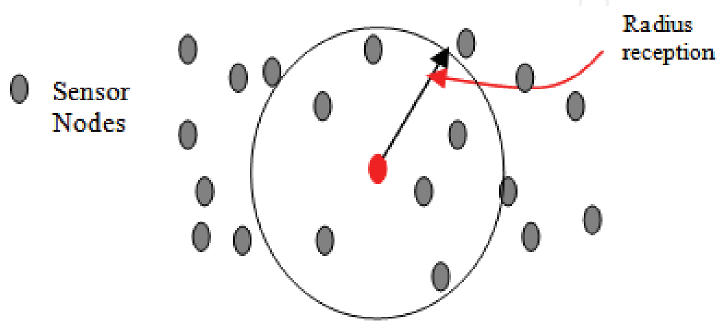


Fig. 1. Radius of reception of a sensor node.

To ensure these aspects, one can find two methods to routing information from a sensor network to a sink:
Reactive (*requested*): If one wants to have the network status at a time *t*, the sink broadcasts a request throughout the network so that sensors send their latest information back to the sink. The information is then forwarded in multi-hop manner to the sink (Shearer, 2008).
Proactive (*periodical*): The network is monitored to detect any changes, sending broadcasts periodically to the sink, following an event in the network such as a sudden change in temperature and movement. Sensors near the event then back the information recorded and route it to the sink (Lu & Sheu, 2007).
Otherwise, depending on the network structure, routing can be divided into flat-based routing, hierarchical-based routing, and location-based routing. All these protocols can be classified into multi-path based, query-based, negotiation-based, QoS-based, or coherent-based routing techniques depending on the protocol operation.
Below, we present some QoS routing protocols:
QoS_AODV (Perkins et al., 2000): The link cost used in this protocol is formulated as a function which takes into account node's energy consumed and error rate. Based on an extended version of Dijkstra's algorithm, the protocol establishes a list of paths with minimal cost. Then, the one with the best cost is selected [10],[13]. This protocol uses hierarchical routing; it divides the network into clusters. Each cluster consists of sensor nodes and one cluster head. The cluster head retrieves data from the cluster and sends it to the sink. QoS routing is done locally in each cluster.
Sequential Assignment Routing (SAR) (Ok et al., 2009): SAR manages multi-paths in a routing table which attempts to achieve energy efficiency and fault tolerance. SAR considers trees of QoS criteria during the exploration of routing paths, the energy resource on each path and the priority level of each packet. By using these trees, multiple paths of the sensors are well trained. One of these paths is chosen depending on energy resources and QoS of path. SAR maintains multiple paths from nodes to sink [9]. High overhead is generated to maintain tables and states at each sensor.
Energy Aware Routing (EAR) (Shah & Rabaey, 2002): EAR is a reactive routing protocol designed to increase the lifetime of sensor networks. Like EDAR, the sink searches and maintains multiple paths to the destinations, and assigns a probability to each of these paths. The probability of a node is set to be inversely proportional to its cost in terms of energy. When a node routes a packet, it chooses one of the available paths according to their probabilities. Therefore, packets can be routed along different paths and the nodes' energy will be fairly consumed among the different nodes. This technique keeps a node from being over-utilized, which would quickly lead to energy starvation.
SPEED (Ok et al., 2009): This protocol provides a time limit beyond which the information is not taken into account. It introduces the concept of "real time" in wireless sensor networks. Its purpose is to ensure a certain speed for each packet in the network. Each application considers the end-to-end delay of packets by dividing the distance from the sink by the speed of packet before deciding to accept or reject a packet. Here, each node maintains information of its neighbors and uses geographic information to find the paths. SPEED can avoid congestion under heavy network load.

3. State dependent routing 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 (Fig 2).

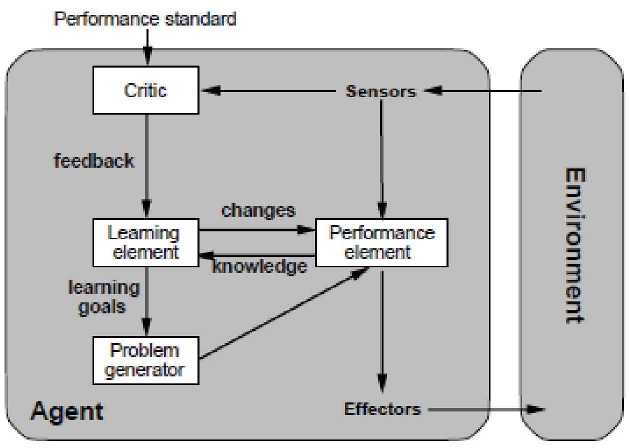


Fig. 2. Agent's learning from the environment.

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 based on RL paradigm:

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.

AntHocNet (Di Caro et al., 2005) is an algorithm for routing in mobile ad hoc networks. It is a hybrid algorithm, which combines reactive route setup with proactive route probing, maintenance and improvement. It does not maintain routes to all possible destinations at all times but only sets up paths when they are needed at the start of a data session. This is done in a reactive route setup phase, where ant agents called reactive forward ants are launched by the source in order to find multiple paths to the destination, and backward ants return to the source to set up the paths in the form of pheromone tables indicating their respective quality. After the route setup, data packets are routed stochastically over the different paths following these pheromone tables. While the data session is going on, the paths are monitored, maintained and improved proactively using different agents, called proactive forward ants.

BeeAdHoc (Wedde et al., 2005) is a new routing algorithm for energy efficient routing in mobile ad hoc networks. This algorithm is inspired by the foraging principles of honey bees. The algorithm utilizes Essentialy two types of agents, scouts and foragers, for doing routing in mobile ad hoc networks. BeeAdHoc is a reactive source routing algorithm and it consumes less energy as compared to existing state-of-theart routing algorithms because it utilizes less control packets to do routing.
KOQRA (Mellouk et al., 2008; Mellouk et al., 2009) 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.

4. AMDR protocol: Adaptive Mean Delay Routing protocol

Our first proposal, called "AMDR" (Adaptive Mean Delay Routing), is based on an adaptive approach using mean delay estimated proactively by each node. AMDR is built around two modules: Delay Estimation Module (DEM) and Adaptive Routing Module (ARM). In order to optimize delay in mobile ad hoc networks, it's necessary to be able to evaluate delay in accurate way. We consider in DEM, the IEEE 802.11 protocol, which is considered to be the most popular MAC protocol in MANET.
DEM calculates proactively mean delay at each node without any packets exchange. The ARM will then exploit mean delay value. It uses two exploration agents to discover best available routes between a given pair of nodes (s, d). Exploration agents gather mean delay information available at each visited node and calculate the overall delay between source and destination. According to delay value gathered, a reinforcement signal is calculated and probabilistic routing tables are updated at each intermediate node using an appropriate reinforcement signal.
ARM ensures an adaptive routing based on estimated mean delay. It combines, on demand approach used at starting phase of exploration process (generation of the first exploration packet for the destination) with proactive approach in order to continue the exploration process even a route is already available for the destination. Proactive exploration keeps AMDR more reactive by having more recent information about network state.

4.1 Delay Estimation Model (DEM)

In order to optimize delay in mobile ad hoc networks, it's necessary to be able to evaluate delay in an accurate way. We propose in this section to estimate one-hop delay in ad hoc networks. It's well known that node delay depends greatly on MAC layer protocol. We consider in our study, the particular case of IEEE 802.11 protocol.
Our Delay Estimation Model (DEM) is based on *Little's theorem* (Little, 1961). Each mobile node, in DEM, is considered as a buffer. Packets arrive to the buffer in *Poisson* distribution with parameter λ. Each mobile node has a single server performing channel access.
Thus, a mobile node is seen as a discrete time M/G/1 queue (Saaty, 1961). According to *Little's* theorem, node delay corresponds to mean response delay defined as fellow:

$$W = \frac{\lambda M_2}{2(1-\sigma)} + b \tag{1}$$

- λ: arriving rate of data packets to the buffer
- b: represents mean service time needed to transmit a data packet with success including retransmission delays
- σ: represents server's rate occupation, it's equal to λb
- M_2: is the second moment of service time distribution.

The generator function of delay service defined in (Meraihi, 2005) is noted *B(z, L,p,k)*, (*L* : packet size, *p :* collision rate and *k*: backoff stage). This generator function can be defined in recursive manner as fellow:

$$B(z,L,p,k) = T_x(z,k,L)\big(1 - p + p * B(z,L,p,2k)\big) \tag{2}$$

$T_x(z,k,L)$ is a transmission delay of a data packet (packet size: L). *B(z, L, p, 2k)* function is invoked in case of collision with a probability *p*. In order to differentiate link's delays, we know that collision rate *p* is different from one neighbor to another. Thus, collision rate becomes a very important parameter for mean delay estimation.

4.2 Adaptive Routing Module (ARM)

AMDR uses two kinds of agents: Forward Exploration Packets (FEP) and Backward Exploration Packets (BEP). Routing in AMDR is determined by simple interactions between forward and backward exploration agents. Forward agents report network delay conditions to the backward ones. ARM consists of three parts:

4.2.1 Neighborhood discovery

Each mobile node has to detect the neighbor nodes with which it has a direct link. For this, each node broadcasts periodically *Hello messages*, containing the list of known neighbors and their link status. The link status can be either symmetric (if communication is possible in both directions) or asymmetric (if communication is possible only in one direction). Thus, *Hello messages* enable each node to detect its one-hop neighbors, as well as its two-hop neighbors (the neighbors of its neighbors). The *Hello messages* are received by all one-hop neighbors, but are forwarded only by a set of nodes calculated by an appropriate optimization-flooding algorithm.

4.2.2 Exploration process

So, FEP agents do not perform any updates of routing tables. They only gather the useful information that will be used to generate backward agents. BEP agents update routing tables of intermediate nodes using new available mean delay information. New probabilities are calculated according to a reinforcement-learning signal.

Forward agent explores the paths of the network, for the first time in reactive manner, but it continues exploration process proactively.

FEP agents create a probability distribution entry at each node for all its delay-MPR neighbours. Backward agents are sent to propagate the information collected by forward agents through the network, and to adjust the routing table entries. Probabilistic routing tables are used in AMDR. Each routing table entry has the following form (Fig. 3):

Dest	(Next$_1$, p$_1$)	(Next$_2$, p$_2$)	…..	(Next$_n$, p$_n$)

Fig. 3. AMDR's routing table entry.

When a new traffic arrives at source node *s*, periodically this node generates a Forward Exploration Packet called FEP. The FEP packet is then sent to the destination in broadcast manner. Each forward agent packet contains the following informations: *Source node address, Destination node address, Next hop address, Stack of visiting nodes, Total_Delay.*
If the entry of the current destination does not exist when the forward agent is created, then a routing table entry is immediately created. The stack field contains the addresses of nodes traversed by forward agent packet and their mean delay. The FEP sending algorithm is the following:

```
Algorithm (Send FEP)
    At Each  T_interval_seconds 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
        Store address of the current node,
        Recover the available mean delay,
        Broadcast FEP,
      Else
        Send BEP
      End If
    End If
End (forward FEP)
```

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 then 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 reception of a FEP packet, the destination node generates a BEP packet which retraces the inverse path of the FEP packet. 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. It's the case of instantaneous delay because the first FEP arriving to destination has the best instantaneous delay. 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: We define an integer *N*, stored at each node, while *N* is positive the destination reply by generating a BEP. It is an intermediate solution between *Reply to all* and *Reply to First* (N=1). The variable *N* is decremented when a BEP is sent. The overhead 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, we compare its *FEP.Total_delay* with the *Node.Total_delay*, and we reply only if the FEP has a delay better or equal to the *Node.Total_delay*. In such manner, we are sure that we adjust routing tables according to the best delay. We have a guarantee of the best delay with reduced overhead.
If the *FEP_Total_delay* is equal or less than a fixed value *D*, the BEP is generated and sent to the source of the FEP. The algorithm of sending a BEP is the following:

```
Algorithm (send BEP)
    Select Case Option
    Case: Reply to All
        Genarate BEP
        BEP.Total_Delay=0,
        BEP.dest = FEP.src,
        Send BEP
    Case: Reply to first
    If (First (FEP) ) Then
        Genarate BEP
        BEP.Total_Delay=0,
        BEP.dest = FEP.src,
         Send BEP
    endIf
    Case: Reply to N
    If (N>0) Then
        Genarate BEP
        BEP.Total_Delay=0,
        BEP.dest = FEP.src,
        Send BEP,
        N=N-1
    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_Delayl <= D) Then
        Genarate BEP
        BEP.Total_Delay=0,
        BEP.dest = FEP.src,
        Send BEP,
    End If
End (Send BEP)
```

Backward Exploration Packet 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_address = BEP.dest) Then
        Update probabilistic routing table
    Else
        Update probabilistic routing table
        Forward BEP
    End If
End (forward BEP)
```

4.3 Reinforcement updating routing tables

Routing tables are updated when a BEP agent is received. This phase can take many forms, and we have chosen updating rules based on (Baras & Mehta, 2003). Since routing table is calculated, data packets are then routed according to the highest probabilities in the routing tables.

Unlike on demand routing protocols, there is no guarantee to route all packets on the same route due to the proactive exploration. A positive reinforcement r^+ will be allotted to node f visited by the BEP agent, when a negative reinforcement r- will be allotted to the all remaining neighbours called n. Let:

- p_{fd}, the last probability that node f choose node d as next neighbour.
- p_{nd}, the last probabilities allotted to the other neighbours ($n \neq f$).
- r, the reinforcement signal which is computed dynamically according to the delay information gathered by the BEP agent.

We assume that D is the delay and μ it's mean (first moment). The value of r indicates the delay quality and is computed according to the following rules:

$$r = \begin{cases} 1/\alpha\left(D/\mu\right) & \text{if } D < \alpha\mu \\ 1, \text{else} & \end{cases} \tag{3}$$

α is a parameter of delay quality defining the interval of delays to be considered as good ones. We apply a correction strategy on r in order to take into account the reliability of D. For example, in the case where α=2, we tend to reinforce more a link when r is less than 0.5, by reducing the r value. In the other case (r> 0.5), the link is penalized by increasing the r value.
The value reinforcement signal r^+ is a positive reinforcement applied to the current node and the punished r^- is a negative reinforcement applied to the remaining MPR neighbours.

$$\begin{aligned} r^+ &= (1-r)*(1-p_{fd}) \\ r^- &= -(1-r)*p_{nd} \end{aligned} \tag{4}$$

BEP makes changes to the probability values at the intermediate and final node according to the following update rules:

$$\begin{aligned} p_{fd} &\leftarrow p_{fd} + r^+ \\ p_{nd} &\leftarrow p_{nd} - r^- \end{aligned} \tag{5}$$

4.4 Flooding Optimization Algorithm

To improve AMDR performance, we introduce a new Flooding Optimization Algorithm (FOA), in order to reduce the overhead generated by the broadcast process. FOA is a delay oriented MPR selection algorithm. Using FOA, we guarantee a reduced overhead generated by exploration agents because FEP agents are forwarded, only, by delay-MPR neighbours selected by FOA.
FOA takes into account the mean delay available at each node. The delay-MPR selection is inspired by bandwidth-MPR algorithm proposed in [9]. Unlike bandwidth-MPR algorithm, FOA defines only one kind of MPRs called delay-MPR. 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.

4.5 Performance evaluation

We use NS-2 simulator to implement and test AMDR protocol. DEM implementation is based on three informations: collision probability τ, channel capacity C and traffic rate λ. Collision probability is calculated using sequence number of ‘Hello’ messages.
We present in the rest of this section three scenarios of simulation. In the first scenario, we define a static topology of 8 nodes. In order to compare AMDR’s performances to both reactive and proactive protocols, we used DOLSR [1] and have extended AODV implementation available on NS-2 in order to be delay oriented routing protocol, noted here QAODV. We have chosen AMDR *Reply to Best* option for comparisons.

Fixed Scenario
The following table summarizes the simulation environment:

Routing	QAODV, AMDR, DOLSR
MAC Layer	802.11
Bandwidth	11Mb/s
TERRAIN	1000m,1000m
Nodes	8
Simulation time	1000 sec
Data traffic	Exponential

Table 1. Simulation settings fixed scenario

At first, in scenario 1, we generate a traffic between nodes '0' and '5'. Few times later, we add a new traffic between node '2' and node '3'.
In a second scenario, we keep the same configuration and we inject simultaneously a third set of traffic between nodes '4' and '1', as showed in figure 4. We compare for each simulation the trace files for each routing protocol.

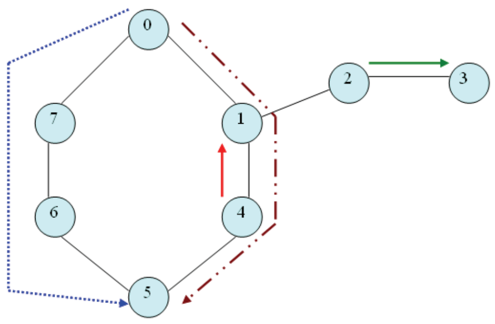


Fig. 4. Topology of fixed scenarios

The comparison of the end-to-end delay realized by QAODV, AMDR and DOLSR protocols is shown on figure 5. We can observe that, at first DOLSR realizes best delays when AMDR and QAODV show a large initial delay, which is necessary for routes to be set up. After initialisation stage, AMDR shows more adaptation to changes in the network load. It realizes the best end-to-end delay followed by QAODV and at last DOLSR.

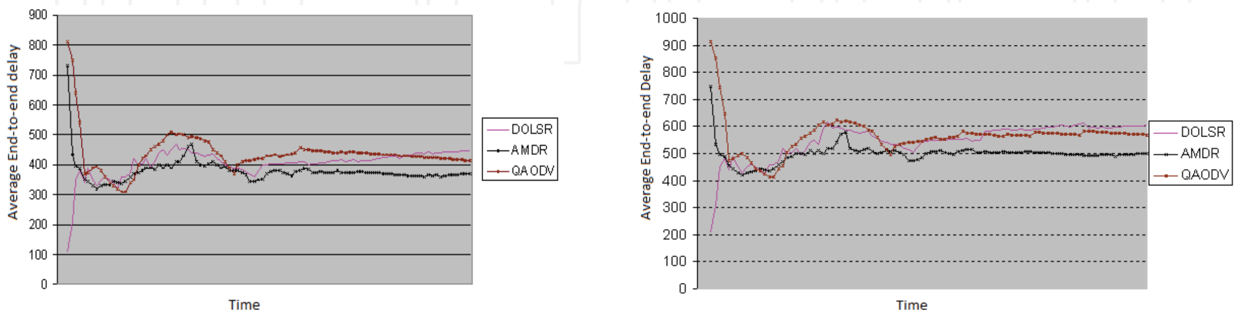


Fig. 5. Packets delay comparison in fixed scenarios (scenario1 & 2)

On the other hand, comparing loss rate performances of the three protocols shows in figure 6, that DOLSR realizes the best performances followed by AMDR and then QAODV.

AMDR performance is justified by keeping alternative paths used when the actual path is broken. Any additional delay is need to route waiting traffics and deliverance ratio is well improved than QAODV. In term of generated overhead, DOLSR was the most important followed by AMDR and QAODV. We explain this by the proactive exploration process used by AMDR, even a route is already established. The difference between overhead of AMDR and QAODV is not very important due to the flooding optimization mechanism used in AMDR.

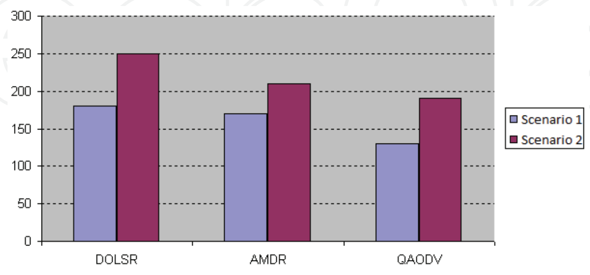


Fig. 6. Loss rate comparison in fixed scenario

Mobility scenario

We study now mobility impact on AMDR and compare its performances with DOLSR and QAODV. We define a random topology of 50 nodes. Table 2 summarizes the simulation setting parameters. We started with a low traffic rate and increase it periodically. After each simulation, we calculate the end-to-end delay realized by each protocol.

Traffic model	Exponential
Surface of simulation	1000m,1000m
Packets size	512 byte
Bandwidth	1Mbs
Mobility rate	5m /s , 10m/s
Number of connections	5, 10, 15, 20, 25
Packets rate	5 packets/s
Simulation duration	500 s

Table 2. Simulation settings scenario 2

Figure 7 summarizes our comparison. We can observe that in low load conditions, there is no difference in end-to-end delays. However, more the network is loaded more AMDR is better in term of delay. Such performance is justified by the easier adaptation of AMDR to changes in the network load.

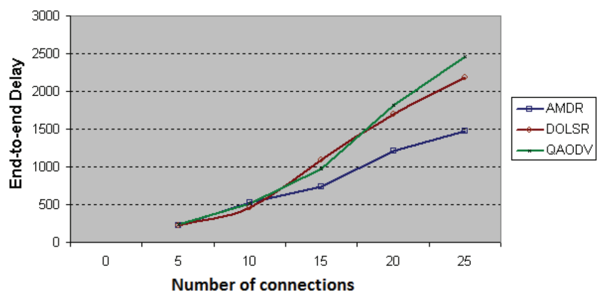


Fig. 7.a Packets delay comparison for low mobility scenario

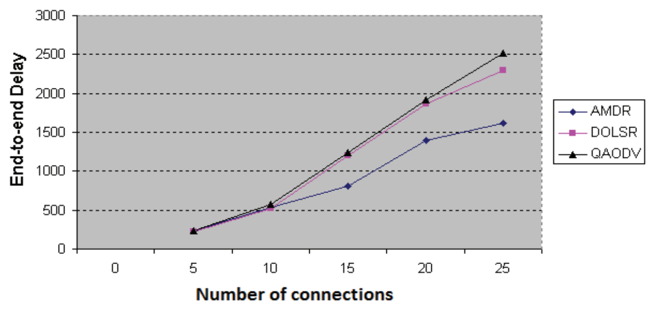


Fig. 7.b Packets delay comparison for high mobility scenario

Comparing loss rate performance between QAODV, AMDR and DOLSR, shows in figure 8 that both AMDR and DOLSR have, in a low loaded network, the same performance when QAODV realises the best performance.
However, in a high loaded network (case of 20 or 25 connections), QAODV becomes less good than AMDR and DOLSR. We justify such results by the adaptation of AMDR to load changes when AODV needs more route request function. We can also observe from mobility scenarios that in high mobility rate conditions, AMDR becomes more interesting than DOLSR and QAODV.

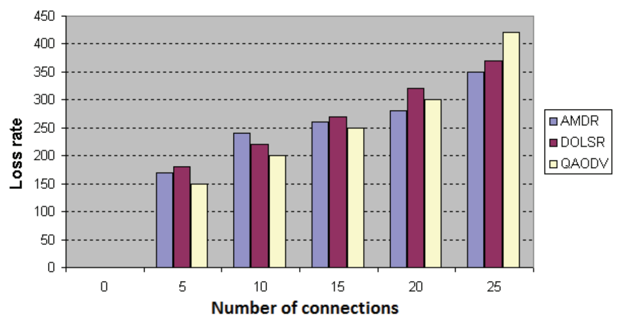


Fig. 8.a Loss rate comparison for low mobility scenario

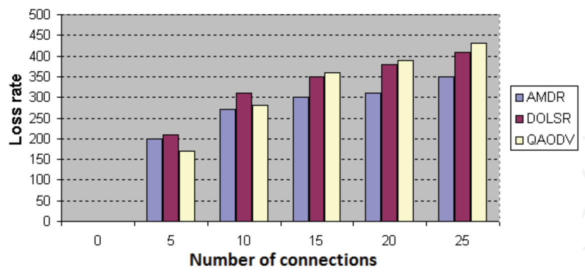


Fig. 8.b Loss rate comparison for high mobility scenario

5. EDAR protocol : Energy and Delay efficient Adaptive Routing protocol

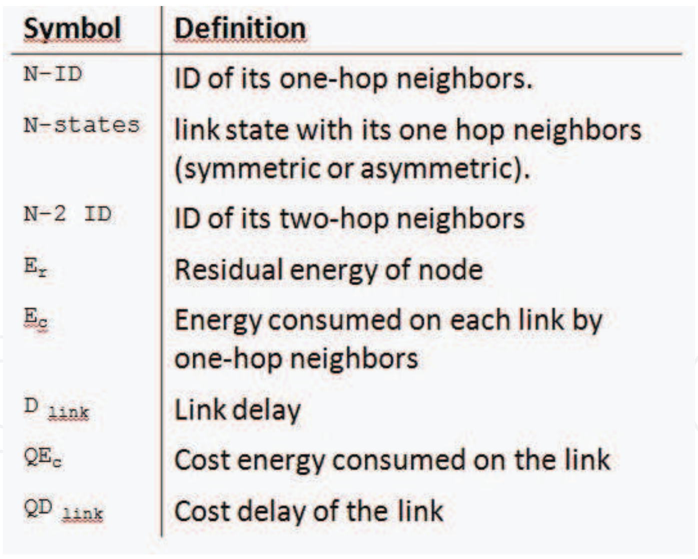
EDAR is our second proposal in this chapter. EDAR is based on adaptive approaches; its objective is to find the best path in terms of energy consumed and end-to-end delay. It assumes that at each node, information about the residual energy, the energy consumed and the average delay links are available even before the route is requested. We assume that these estimates are made during the discovery of neighbors by an external mechanism independent of routing process.

Our proposal is also considered as a hybrid protocol; it combines the concept of on-demand route search, and the proactive exploration concept. This joint mechanism in the exploration phase allows to EDAR to find alternative routes missed in the first phase. For this, EDAR is based on two explorer agents: *Int-E-Route* (Interest Exploration Route) and *Resp-E-Route* (Response Exploration Route). The first one, generated at the sink when this one demands a request, is available in the network to find routes to the source node. To limit overhead generated by route discovery, we use an optimized distribution mechanism based on *Multi Point Relay* (MPR) OLSR protocol (Nguyen & Minet, 2007).
Of these two explorer agents, one is sent by the sink to a node, and the other is sent by the node in response to the first one. The arrival of an *Int-E-Route* agent to the source node initiates the creation of the second agent, *Resp-E-Route*. This latter takes the reverse path followed by the first *Int-E-Route* agent.

5.1 EDAR's agents

5.1.1 Int-E-Route agent

We use the mechanism of periodic "Hello" messages to discover neighbors. Each node broadcasts Hello messages, containing a list of its neighbors and the links state with its symmetrical neighbors, asymmetrical or lost. "Hello" messages are received by its one-hop neighbors and will not be relayed; each node is able to know its one- and two-hop neighbors. A message is sent in broadcast and will not be relayed (the distribution is local). This latter is composed by: (1) a list of addresses of its symmetrical neighbors, and (2) a list of addresses of its asymmetrical neighbors. There is a neighbor table in each node of the network. Fig. 9 describes the content of this table.



Symbol	Definition
N-ID	ID of its one-hop neighbors.
N-states	link state with its one hop neighbors (symmetric or asymmetric).
N-2 ID	ID of its two-hop neighbors
E_r	Residual energy of node
E_c	Energy consumed on each link by one-hop neighbors
D_{link}	Link delay
QE_c	Cost energy consumed on the link
QD_{link}	Cost delay of the link

Fig. 9. EDAR's Neighbors table.

When a "Hello" message is received by a neighbor, if a node finds its own address in the asymmetrical list, then it declares that this link is valid. A node is considered as a neighbor if and only if the collision rate of "Hello" packets sent to it, is below a certain threshold fixed initially.
Whenever *Int-E-Route* gets through an intermediate node, it retrieves its address and the information of its residual energy, energy consumed on the link and the average delay. *Int-*

E-Route packets are sent periodically as the route to this node is requested. This period is defined by the exploration interval which is calculated by simulation (for our scenarios, this value started with 10 seconds). Thus, at each iteration, a new packet *Int-E-Route* with a new sequence number is created and sent in the same manner as the first *Int-E-route*.
This agent is sent in broadcast to all one-hop neighbors nodes. To reduce the overhead generated by route exploration, only MPRE nodes will be allowed to relay this packet.
Int-E-Route recovers energy consumed and delay with its neighbors and stores it in its memory. The sequence number of the packet does not change. This process continues until *Int-E-Route* arrives to the source node.
The cost of the path from the source to sink is calculated as follows.
Let the minimal residual energy of each node be a parameter fixed beforehand (it corresponds to the energy required to send a fixed volume of information):

$$E_r \geq \varepsilon \tag{6}$$

where E_r represents the residual energy and the threshold of minimum energy for all active nodes before sending all data.
The cost of a node *i* for a link between nodes *i* and *j* is computed as:

$$\text{Cost of node}_i^j = \text{Cost of link}_i^j = \gamma QE_{C_i}^j + \theta QD_i^j \tag{7}$$

where:

$$f_{\text{cost}} = \sum_{j=1}^{K} h_j C_j \tag{8}$$

- QE_c is the cost of energy consumed on link (*i*, *j*),
- *QD* represents the cost of mean delay on link (*i*, *j*),
- γ and θ are tuneable parameters, with $\gamma \succ \theta$ to give more importance to the energy metric.

The cost function can be completed as the cost of path as follows:
C is a scalable vector for QoS metrics, and *K* represents the number of QoS criteria considered in the routing process.
The cost of the whole path constructed with N nodes between the source node and the sink is:

$$\text{Cost of path} = \gamma \sum_{i=1}^{N-1} QE_{C_i} + \theta \sum_{i=1}^{N-1} QD_i \tag{9}$$

5.1.2 EDAR's exploration mechanism

In our EDAR proposal, we used the same mechanism as OLSR (Nguyen & Minet, 2007by replacing the cost function by the energy cost stored in the neighbors table (fig. 9). So, the choice of MPRE for each node will be based on the links that offer the best cost estimated locally. The algorithm for selecting MPRE should be summarized as follows:

- Each node N_i of the network as MPRE chooses among its neighbors for one-hop, all the nodes up to a neighbor with two-hops.

- Each node selects as MPRE node that has the best link cost. In the case of equality between two nodes, the one that covers more than two-hop neighbors is chosen. This step is repeated until all neighbors at two hops are covered.

5.1.3 EDAR's Resp-E-Route agent

Agent *Resp-E-Route* updates routing tables based on information in the neighbors tables over all nodes crossed. This update consists of readjustment of link costs associated with each entry in routing table and generated by source node upon reception of *Int-E-Route* packet. It retrieves all information gathered by the packet *Int-E-Route* at the sink. The Packet *Resp-E-Route* is sent in unicast mode and takes the opposite route borrowed by the packet *Int-E-Route*. The source node generates packets *Resp-E-Route* on the basis of information on total cost of path followed by the packets *Int-E-Route*. When the arrival of the first packet *Int-E-Route* to source node is happen, the variable *"best cost path"* is used to store the cost of the path. At each arrival *Int-E-Route* packet to the source node, the cost of path calculated for this route is compared with the lowest cost path saved in the variable "best cost route". If there is a better path, the traffic will switch to this one. Each *Resp-E-Route* packet retraces exactly the same path taken by the *Int-E-route* packet which is at the origin of the path. At each passage trough the intermediate node, it compares its address with the destination address of sink *Resp-E-Route* packet to verify if it reaches the destination.

5.1.4 EDAR's updating and calculating routing table

Routing process consist to find the path with minimum cost in order to minimize simultaneously node energy consumed on all paths and the end-to-end delay between sink and source node. The fundamental problem is to update the link cost; the algorithm that we propose is adaptive and is based on routing tables maintained at each node. When a route is requested, the node controls its routing table if there is an entry that corresponds to requested node. If no entry appears for that node, the node creates an entry and initializes the cost of links; evenly distributed over all its neighbors MPRE.
These costs are updated during transition from the agent *Resp-E-Route* based on information of links cost retrieved by visiting all nodes on the route. The link costs are updated as follows:

$$Cost\ of\ node_i^j = Cost\ of\ link_i^j = \gamma QE_c + \theta QD \tag{10}$$

The new costs at $(t+\delta t)$ in the node i on the link (i, j) is:

$$QE_{c_i}^j(t+1) = \frac{E_{c_i}^j(t)}{E_{c_i}^j(t+\delta t)} * QE_{c_i}^j(t) \tag{11}$$

and

$$QD_{c_i}^j(t+1) = \frac{D_{c_i}^j(t)}{D_{c_i}^j(t+\delta t)} * QD_{c_i}^j(t) \tag{12}$$

Where:
$E_{c_i}^j(t)$: Energy consumption in the node *i* on the link *(i, j)* at a time t.

$D_{c_i}^{j}(t)$: Link delay in the node *i* on the link *(i, j)* at a time t.

We refresh then the new cost of the path based on equation (4) and continues iteratively the update process.
Finally, the variable "Best cost route" is updated as follow:

$$\text{Best cost route} = \text{Max}\left(\gamma \sum_{i=1}^{N-1} QE_{c_i} + \theta \sum_{i=1}^{N-1} QD_i\right) \tag{13}$$

5.2 Performance evaluation

This section first describes our simulation setup and the metrics we used to evaluate EDAR. Then, it presents the results obtained under three scenarios designed to compare EDAR with three other protocols (section 2): QoS_AODV, SAR and EAR. In the case of QoS_AODV, we have modified the original algorithm in order to use the energy metric for the choice of the paths. Our extension takes into account the end-to-end delay and replaces the bandwidth metric with the energy consumed along the path.

5.2.1 Simulation setup and metrics

We use NS-2 for our simulations. We randomly distribute 100 nodes on a 200x200m area. The nodes send 35 bytes messages at a maximum of 200 kbps. The initial energy for each node is 5J. For each scenario, we run 10 simulations with a topology chosen at random. Each run lasts 300 seconds. Finally, we set γ to 0.75 and θ to 0.25, to give more importance to the energy metric.
For the evaluation, we stress the four protocols (QoS_AODV, SAR and EAR and EDAR) by varying mobility network conditions. Varying each network parameter allows to study a different feature of the routing algorithms. Highly mobile nodes will test the algorithm's ability to quickly detect route failures and establish new ones. For each mobility scenario, we increase one of these network parameters to force network congestion and energy starvation. We expect that the different protocols will react and behave differently to protect the network's lifetime, while guaranteeing a good QoS.
To this end, we are interested in three metrics to evaluate EDAR and compare it to the other protocols. These metrics were carefully chosen in order to give a broad idea of the improvements achieved by our protocol:

- The *Energy consumption* represents the energy (in mJ) spent by the nodes to route the packets to the sink. It includes both data packets and signalling packet. Energy consumption has a direct impact on the lifetime of a sensor network.
- The *End-to-end delay* is the time for a packet to reach the sink. It includes the time spent to discover the routes, as well as the time spent in the nodes queues and in the air. This metric represents the global response time of the network, and depends on the protocol efficiency is terms of the number of congested nodes and stability.
- The *Delivery rate* is the number of lost packets divided by the number of packets sent. This metric represents the reliability of the protocol in terms of packet delivery.

5.2.2 Results

Our study concerns the evaluation of the protocols for an increasing mobility of the deployed sensors. The first case shows the protocol's behaviour when nodes are completely static (0 m/s), and we consider extreme cases where nodes move at up to 30 m/s.

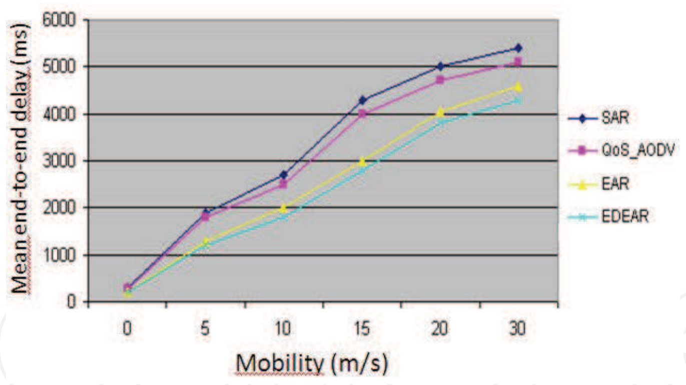


Fig. 9. Mean delay for various mobility patterns

Fig. 9 plots the mean end-to-end delay achieved by all four protocols under various speeds of the nodes. In the case of a static topology, the results for all protocols are quite similar. However, increasing the mobility reveals a clear difference in the protocols' efficiency. The plot indicates that, for speeds above 5 m/s, the performances of Qos_AODV and SAR degrade quickly (the delay increases from 2000 to 5000 ms), while EAR and EDAR keep delivering packets on a reasonable delay, for a speed up to 10 ms/s. Still, our protocol resists better to increasingly harder conditions.

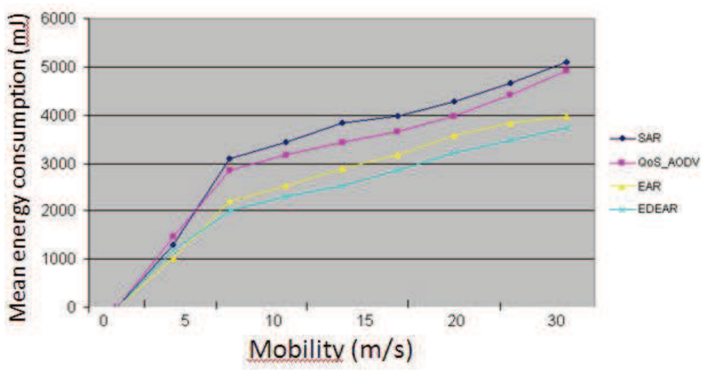


Fig. 10. Mean energy consumption for various mobility patterns

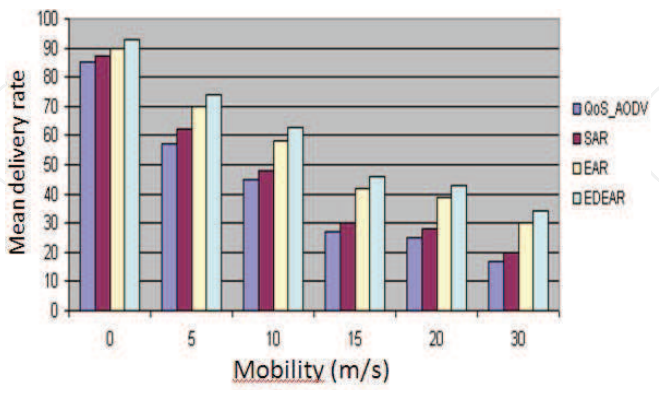


Fig. 11. Mean delivery rate for various mobility patterns

When nodes are mobile, topology changes are frequent. These changes cause a frequent loss of the currently selected best paths, therefore triggering route discovery phases more frequently. Discovery phases generate additional control traffic, consuming more energy and bandwidth. As a result, resources for data traffic are reduced, and the nodes quickly

consume their energy. To study this effect, Fig. 10 plots the energy consumption when nodes move at various speeds. The dynamics clearly have less impact on EDAR, which takes advantage of its improved reactive capabilities when compared to the other protocols. Indeed, the choice of both energy-saving and low delay nodes dramatically reduces the energy consumption. Routing tables are probabilistic in EDAR; several paths may be selected to reach a destination. Therefore, if a node's neighbours change, the mechanism of adjusting probabilities can be adapted almost instantly to degradation performance on the selected path. Furthermore, nodes mobility can lead to the creation of new routes involving nodes with a high residual energy. EDAR quickly detects and selects these high quality nodes, thus allowing it to resist more efficiently to frequent route changes.
Finally, Fig.11 considers the mean delivery rate achieved when the mobility increases. Under a static scenario, all protocols achieve a very good delivery rate, EDAR being the only one higher than 90%. As expected, the delivery rate dramatically decreases with an increasing mobility, but the decrease is less significant for adaptive protocols, especially for EDAR. For a speed of 10 m/s, EDAR still achieves a delivery rate greater than 60%, while SAR and Qos_AODV are well below 50%. Packet losses occur more frequently as the network dynamicity increases, because routing protocols have to face the problem of recomputing new routes all the time. During this phase, no routes are known to reach the destination, and the nodes have no choice but dropping the packets. However, thanks to EDAR's exploration process and its probabilistic nature, it is able to quickly reallocate a new route, thus decreasing the time when no route is available.

6. Conclusion

QoS management in networking has been a topic of extensive research in a last decade. As the Internet network is managed on a best effort packet routing, QoS assurance has always been an open issue. Because the majority of past Internet applications (email, web browsing, etc.) do not used strong QoS needs, this issue is somewhat made less urgent in the past. Today, with the development of network real-time application and the convergence of voice and data over heterogeneous networks, it is necessary to develop a high quality control mechanism to check the network traffic load and ensure QoS requirements. 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). Therefore, with the wide emergence of real time applications in mobile and sensor networks, QoS guarantees become increasingly required. Therefore, protocols designed for MANETs and WSNs should be involved in satisfying application requirements while optimizing network resources use. A considerable amount of research has been done recently in order to ensure QoS routing in mobile ad hoc networks. Several multi-hop routing protocols have been developed. In addition to proactive, reactive and hybrid approaches, often used by the most known routing protocols, adaptive routing protocols which are by nature oriented QoS have already proved their success in wired networks. They are based on reinforcement learning mechanisms and mainly built around exploration agents having for task to discover routes and gather information about the state of visited links. However, for a network node to be able to make an optimal routing decision, according to relevant performance criteria, it requires not only up-to-date and complete knowledge of the state of the entire network but also an accurate prediction of the network dynamics during propagation of the message through the network. This problem is naturally formulated as a dynamic programming problem, which, however, is too complex to be solved exactly. Reinforcement learning (RL) is used to

approximate the value function of dynamic programming. In these algorithms, the environment is modeled as stochastic, so routing algorithms can take into account the dynamics of the network. However no model of dynamics is assumed to be given.
In this chapter, we have focused in first part our attention in some special kind of Constrained Based Routing in mobile networks which we called Q○S self-optimization Routing. It is shown from simulation results that combining proactive exploration agents with the on-demand route discovery mechanism, our AMDR routing protocol would give reduced end-to-end delay and route discovery latency with high connectivity. This is ensured due to the availability of alternative routes in our algorithm. The alone case where our approach can provide more important delay is the first connection where any route is yet established. On the other hand, the use of delay-MPR mechanism, guarantees that the overhead generated will be reduced. From simulation results we can estimate that AMDR realizes best performance than adaptive routing protocols based on swarm intelligence using instantaneous delay.
Secondary, we study the use of reinforcement leaning in EDAR protocol in the case of Wireless Sensor Networks. We presented a new way to make adaptive routing with quality of service in order to improve the end-to-end delay and increase the lifetime of a delay tolerant network. Our protocol, called EDAR, explores the network and chooses the best path routing in terms of energy and delay to route information based on reinforcement learning paradigm. For that, an explorer agent is used and learns from past experiences to choose the next path; data is always sent over the best path. Our approach offers advantages compared with other classical approaches. In particular, it reduces much better the energy consumed required for information sent, updates routing and network exploration more reactively. The proposed algorithm takes into account the network state in a better way than the classical approaches do. Also, our protocol is well-suited for a dynamic mobile environment., especially in Delay Tolerant Networks.
Finally, extensions of the framework for using these techniques across hybrid networks to achieve end-to-end QoS needs to be investigated, in particular on large scalable networks. Another challenging area concerns the composite metric used in routing packets (especially residual bandwidth) which is so complex and the conditioning of different models in order to take into account other parameters like the information type of each flow packet (real-time, VBR, …).
In Final, we show in this chapter that an adaptive algorithm based on RL that simultaneously processes an observation and learns to perform better is a good way to consider time evolving networks. Such adaptive algorithms are very useful in tracking a phenomenon that evolves over time.

7. Acknowledgments

The work presented here is a part of UPEC-LISSI research activities team, especially in the scope of the supervising Nesrine Ouferhat, Saida Ziane PhD's thesis and discussions with Dr. Brice Augustin and Dr. Said Hoceini. I would like to thank them for their support and their suggestions.

8. References

Baras, J.S., Mehta, H. (2003). A Probabilistic Emergent Routing Algorithm (PERA) for Mobile Ad Hoc Networks, *Proceedings of WiOpt '03: Modeling and Optimization in Mobile, AdHoc and Wireless Networks*, Sophia-Antipolis, France.

Boyan, J. A., Littman, M. L., (1994). Packet routing in dynamically changing networks: A reinforcement learning approach, *Advances in Neural Information Processing Systems 6*, Morgan Kaufmann, San Francisco, CA, pp. 671-678.

Di Caro G., Ducatelle F., Gambardella M., (2005). "AntHocNet: An Adaptive Nature-Inspired Algorithm for Routing in Mobile Ad Hoc Networks". *European Transactions on Telecommunications (ETT), Special Issue on Self Organization in Mobile Networking*, Volume 16, Number 5, Pages 443-455.

Dorigo, M., Stüzle, T., (2004). *Ant Colony Optimization*, MIT Press, Cambridge, MA.

Gallager, R.G. (1977). A minimum delay routing algorithm using distributed computations. *IEEE Transactions on Communications*, 25(1), 73-85.

Gelenbe, E., Lent, L., Xu, Z, (2002). Networking with Cognitive Packets, Proc. ICANN 2002, Madrid, Spain, pp. 27-30.

Little, J.DC. (1961). *A proof of the Queueing Formula L=λW*, Operations Research.

Lu Y.J., Sheu T.L., (2007). "An efficient routing scheme with optimal power control in wireless multi-hop sensor networks", *Computer Communications*, vol. 30, no. 14-15, pp. 2735-2743.

Mellouk, A., Lorenz, P., Boukerche, A., Lee, M.H., (2007). Impact of Adaptive Quality of Service Based Routing Algorithms in the next generation heterogeneous networks, *IEEE Communication Magazine*, IEEE Press, vol. 45, n°2, pp. 65-66.

Mellouk, A., Hoceini, S., Cheurfa, M., (2008). Reinforcing Probabilistic Selective Quality of service Routes in Dynamic Heterogeneous Networks, *Computer Communication Journal*, Elsevier Ed., Vol. 31, n°11, pp. 2706-2715.

Mellouk, A., S. Hoceini, S. Zeadally, (2009). Design and performance analysis of an inductive QoS routing algorithm, *Computer Communications Journal*, Elsevier Ed., Volume: 32 n°12, pp. 1371-1376, Elsevier, 2009.

Naimi Meraihi A., (2005). *802.11 AdHoc Networks Delay and Routing*, PhD Thesis, INRIA Rocquencourt, France.

Nguyen D-Q., Minet P., (2007). "Analysis of MPR selection in the OLSR protocol". *Proc. Of PAEWN*, Niagara Falls, Ontario, Canada.

Ok C., Lee S., Mitra P., Kumara S., (2009). "Distributed energy balanced routing for wireless sensor networks", *Computers & Industrial Engineering*, vol. 57, no. 1, pp. 125-135.

Ozdaglar, A.E., Bertsekas, D.P. (2003). Optimal Solution of Integer Multicommodity Flow Problem with Application in Optical Networks. *Proc. Of Symposium on Global Optimisation*, June, 411-435.

Perkins C.E., Royer. S, Das R., (2000). "Quality of Service for Ad hoc On-Demand Distance Vector Routing", *IETF Internet Draft*.

Saaty T.L., (1961). *Elements of Queueing Theory and its applications*, Mac Graw-Hill Ed.

Shah R.C., Rabaey J.M., (2002). "Energy aware routing for low energy ad hoc sensor networks". *Proceeding of IEEE Wireless Communications and Networking Conference* (WCNC), Orlando, FL, USA.

Shearer F., (2008). "System-Level Approach to Energy Conservation", *Power Management in Mobile Devices*, pp. 217-259.

Sutton, R.S., Barto, A.G. (1997). *Reinforcement Learning*. Ed. MIT Press.

Wedde H.F., Farooq M., Pannenbaecker T., Vogel B., Mueller C., Meth J., Jeruschkat R., (2005). "BeeAdHoc: an energy efficient routing algorithm for mobile ad hoc networks inspired by bee behavior", In *Genetic And Evolutionary Computation Conference Proceedings,* pp 153-160.