

Toward Neural Networks Solution for Multimedia Support in Mobile Ad hoc Networks

Lyes Khoukhi and Soumaya Cherkaoui

Department of Electrical and Computer Engineering, University of Sherbrooke
J1K 2R1, QC, Canada

{Lyes.Khoukhi, Soumaya.Cherkaoui}@USherbrooke.ca

Abstract—In this paper, we detail an integrated model for quality of service (QoS) support with service differentiation based on neural networks in wireless mobile ad hoc networks. The model aims at minimizing end-to-end delay for real-time traffic when the network has low to medium dynamics. The use of multi-service classes allows per-flow end-to-end QoS support. The operation of the model is based on a set of core functionality that is augmented with an intelligent learning layer. Core operations ensure different tasks of routing and QoS support control such as route discovery, resources reservation, admission control, traffic adaptation and detection and recovery of QoS violation. The added intelligent layer learns from the different operations performed by the core functionality by means of a Multilayered Feedforward Neural Network according to user requirements and network state. The integrated model showed in simulations that the core functionality together with the intelligent learning improve the quality of data delivery in the network comparatively to models such as SWAN. The effect of the learning rate on the convergence of network training was also examined. Using intelligent neural networks tools to support the multimedia services in ad hoc networks is a promising avenue.

Index Terms—Quality of Service, Service Differentiation, Wireless Mobile Ad hoc Networks, Neural Networks.

I. INTRODUCTION

There is a growing need to support diverse kinds of multimedia applications in the wireless networks. These applications, such as video and audio real-time traffic, usually require time-bounded service and bandwidth guarantee. Therefore, there is a vital need to provide quality of service (QoS) support in order to assure better quality delivery. However, providing QoS support for the delivery of real-time audio, video and data in wireless mobile ad hoc networks presents a number of significant technical challenges:

First, the scarcity of resources in ad hoc networks requires that the QoS mechanisms and signaling do not exhaust these resources. Second, the interconnection between source-destination devices relies on peer wireless and mobile nodes that operate as routers. Then, rerouting among mobile nodes causes topology and

network load conditions to change dynamically, which may complicate the support of real-time applications with appropriate QoS. Third, the allocation of resources for flows must occur rapidly with minimum overhead, while reacting adequately to frequent node mobility. Fourth, the dynamic nature of ad hoc networks makes difficult to dynamically assign central controller to maintain connection state and reservations [1]. Fifth, flow relying on a pre-established path and resource reservation along the path will suffer from traffic interruptions due to the frequent path changes. Sixth, QoS requirements, such as latency, are time-variant and can considerably change in a short time due to the nodes mobility and radio interferences.

In response to these challenges, there is a vital need to design new QoS approaches for supporting the multimedia services. These approaches need, for instance, to be capable of differentiating between various service requirements of user sessions (e.g., continuous media flow, etc.). Furthermore, other essential elements should be considered carefully when developing new QoS approaches; such as traffic adaptation and congestion management, detection and recovery of QoS violation. The ultimate goal to be reached is the guaranty of a best end-to-end service quality delivery

In this paper, we explore the efficiency of the artificial intelligent neural networks tool for the QoS support of multimedia applications. For that aim, we introduce NNMS, an intelligent QoS model with service differentiation based on neural networks for wireless mobile ad hoc networks. Our model is composed of: a) kernel plan which assures the basic functions of routing and QoS support control such as: route discovery and resources reservation, admission control, traffic adaptation, detection and recovery of QoS violation, and b) intelligent learning plan which assures the learning of kernel operations using neural networks tool. In our proposal, we implemented the multilayer feedforward neural network for optimizing the kernel decision adaptively according to the network state. Note that the power and usefulness of neural networks have been demonstrated in several applications including speech synthesis, diagnostic problems, medicine, business and finance, robotic control, computer vision, etc... For some application areas, neural models show promise in

achieving human-like performance over more traditional artificial intelligence techniques.

In order to be conform to the third generation wireless telecommunication systems (3G), several service classes are introduced in the NNMS kernel. These traffic classes allow a wide range of requirements as defined by 3G: interactive, conversational, streaming, etc. One of the advantages of our NNMS kernel plan is that the resources allocated during the reservation phase are automatically removed in an independent and fully-distributed manner when a flow path changes due to dynamic topology. In addition, the QoS violation caused by the dynamic topology at lower and medium mobility is recovered. Furthermore, Multilayered Feedforward Neural Network (MFNN) allows a fast learning of different operations performed by the kernel and permits a best traffic class selection and delivery according to flow requirements.

The structure of the paper is organized as follows: We present the related works in Section II. An overview of our QoS model is presented in Section III. Section IV describes the main mechanisms and functionalities of the kernel. Section V presents the application of neural networks on the kernel. The simulation results are shown in Section VI. Finally, Section VII concludes the paper.

II. RELATED WORKS

Many researches have focused on addressing the QoS issue in ad hoc networks. SWAN [1], INSIGNIA [2] and FQMM [3] are the most noteworthy QoS models attempting to establish comprehensive solutions. These models provide some good insight into QoS issues in the ad hoc environment. FQMM is a hybrid approach combining the advantages of per-class granularity of DiffServ with the per-flow granularity of IntServ. It tries to preserve the per-flow granularity for a small portion of traffic, given that a large amount of the traffic belongs to per aggregate of flows, that is, per-class granularity. FQMM offers a good solution for small- and medium-size ad hoc network, but it is not suitable for large networks.

INSIGNIA is one of the noteworthy QoS frameworks with per-flow granularity and reasonable treatment for mobility. INSIGNIA employs an in-band signaling system that supports fast reservation, restoration, and adaptation algorithms. Three levels of services are implemented: best effort, minimum, and maximum. However, the bandwidth is the only QoS parameter used in INSIGNIA. SWAN proposes a service differentiation in stateless wireless ad hoc networks using distributed control algorithms. The feedback delay received from the MAC layer is exploited as a measure of network congestion. SWAN uses mechanisms of rate control and source-based admission control in order to promote the rate control system used at each node to treat traffic either as real-time or best-effort traffic. However, one of the drawbacks of SWAN is how to calculate the

threshold rate limiting any excessive delay that might be experienced [5]. It also uses merely two levels of services: real-time and best effort traffic. SWAN and INSIGNIA are intranet QoS models providing services that have to be mapped to either per-flow or per-class services, but SWAN remains the best example of stateless distributed QoS framework developed for wireless ad hoc networks.

On the other hand, several works [6]-[14] have studied the QoS routing issue. In [6], we have proposed a flexible QoS routing protocol (AQOPC) based on multi-service classes and multi-path scheme. It provides information about the state of bandwidth, end-to-end delay and hop count in the network. AQOPC performs an accurate admission control and optimal use of network resources by calculating multiple paths and generating the needed service classes to support different QoS user requirements. Lin and Liu [7] have addressed the problem of supporting real-time communications in a multihop mobile network using QoS routing that permits bandwidth calculation and slot reservation. This protocol can be applied to two important scenarios: multimedia ad hoc wireless networks and multihop extension wireless ATM networks. Xue and Ganz [8] introduced the ad hoc QoS on-demand routing (AQOR), which integrates signaling functions for resource reservation and QoS maintenance at per-flow granularity.

Some works [7], [12] have proposed table-driven routing approaches for QoS support. However, their performances are low compared to reactive approaches, due to the problem of stale route information [13]. A link-state QoS routing protocol for ad hoc networks (QOLSR) was proposed in [9] in the aim of implementing QoS functionality to deal with limited available resources in a dynamic environment. A ticket-based QoS routing protocol was proposed in [10]. This protocol is based on a model which assumes that the bandwidth of a link can be determined independently of its neighboring links. Using the same model, [11] proposes a QoS multi-path routing protocol based on a ticket-distribution scheme to satisfy bandwidth constraints. In terms of routing decision, the previous protocols could be classified into either source routing or distributed routing schemes. The source routing schemes such as [13] suffer from problems of scalability and frequent updates of the state of the network. On the other hand, most of the existing distributed algorithms (e.g., [14]) require the maintaining of a global network state at every node, which may cause the scalability problem.

Note that the ability to provide QoS depends also on how well the resources are managed at the MAC layer. Some among the works cited above used generic QoS measures and are not tuned to a particular MAC layer [2], [10]. Some others use CDMA to eliminate the interference between different transmissions [7], [15]. Chen *et al.* in [16] introduced an on-demand, link-state, multi-path QoS routing protocol which collects information of link bandwidth from source to destination

under the CDMA-over-TDMA channel model. Similarly, CDMA-over-TDMA channel model has been adopted in [15] using the notion of a time slot on a link to calculate the end-to-end path bandwidth. The same model has been used for calculating the end-to-end path bandwidth to develop on-demand QoS routing [7] and DSVD based QoS routing [15].

III. NNMS OVERVIEW

We can view NNMS as a system composed of a kernel coupled with an intelligent learning algorithm. Fig. 1 shows the NNMS plans: routing and QoS support control plan, and intelligent learning plan. The cooperation between the two plans would improve the quality of data delivery in the network. The first plan is based on multi-service classes that provide per-flow end-to-end QoS support. The intelligent learning plan permits a dynamic network adaptation to the traffic conditions using an intelligent neural network tool. This tool allows, once the learning of NNMS kernel operations is achieved, a decrease of the time required to perform the kernel tasks.

Fig. 1 shows the main functionalities of NNMS: the routing scheme allows discovering the existing routes between source and destination pairs. In our study, we considered an on-demand routing scheme because of its flexibility and advantages in comparison to proactive scheme. The QoS support control sub plan is responsible for providing QoS guaranties to the user requirements in terms of different constraints (bandwidth, delay, hop count). Thus, the required functionalities, such as resources reservation and admission control, are assured by this plan. The guaranty intervals service classes (GISC) are a set of intervals that represent the network state in terms of different metrics. GISC allows a suitable path selection according to the flow requirement. The update of GISC is performed at any variation in network resources or recovery of QoS violation. The detection and recovery of QoS violation is assured by two new techniques that we have developed and integrated within the kernel.

The neural networks perform the training of the kernel operations and the testing of set of QoS requirements in order to verify the learning level of NNMS. Once the training phase is achieved, the testing and validation phases will be realized. Then, a given node would be able to select the next hop for routing the received data packets without performing any other treatment such as route discovery. In the following, we start by presenting the different tasks of the kernel, and then we explain the intelligent learning plan based on neural network algorithm.

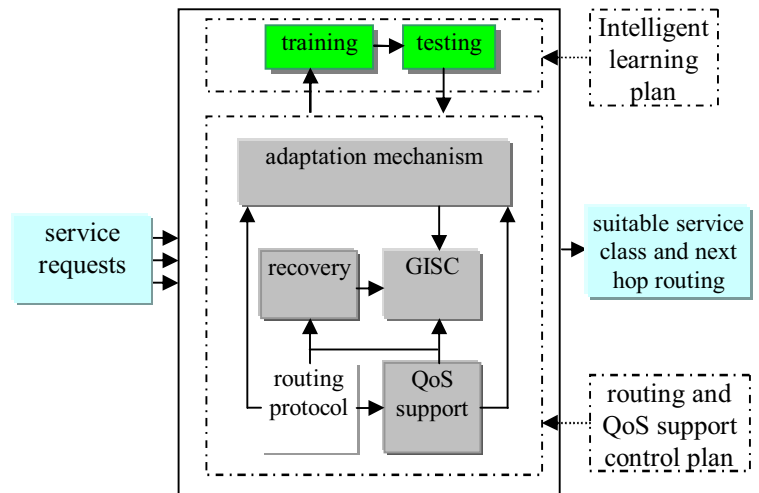


Figure 1. NNMS global view

IV. NNMS KERNEL

The schematic diagram of NNMS kernel is illustrated in Fig. 2. The multi-criteria admission control efficiently estimates the local available bandwidth at each node, and provides accurate information about the state of bandwidth, end-to-end delay and hop count of routes discovered between source and destination. The decision to admit a new flow is done by this mechanism. The routing scheme performs the discovery of routes and the reservation of resources. The classifier is able to differentiate between flows in terms of QoS requirements: delay, bandwidth, and best effort flows. In order to regulate traffic, priority queuing is used so that traffic belonging to lower priority classes is regulated more aggressively than those belonging to higher priority classes. The guaranty intervals are dynamically adjusted according to the feedback information received by the node concerning the network state. Note that in this paper, we consider only low and medium network mobility. More details about NNMS kernel mechanisms will be provided in the following sections.

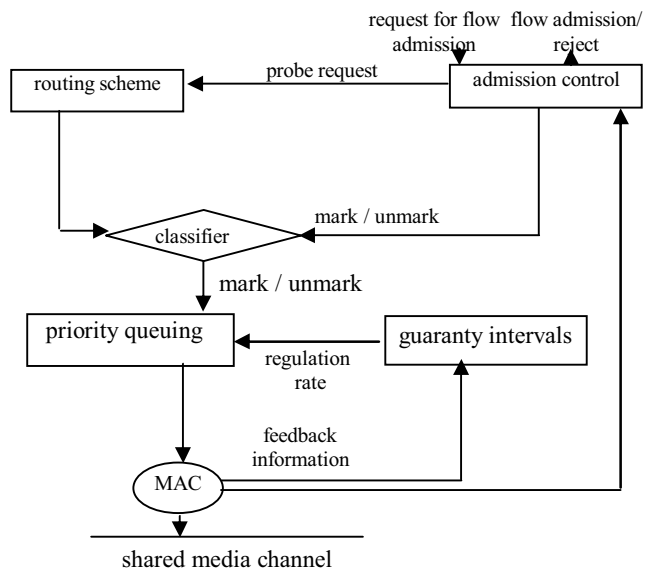


Figure 2. NNMS kernel model

A. Routing Scheme

The main operations of the routing scheme are path discovery and resources reservation. A node state in this scheme may take one of the following states: *checked* during the path discovery, *reserved* during the resources reservation, *allocated* at the reception of data packets, *liberated* once the resources are not used by the node. These states are useful for the detection and recovery of QoS violation as will be explained in Section IV.E. In addition, the resources reserved are automatically liberated to be used by other flows if no data packets are received within a short period. The routing scheme follows a pure on-demand rule (i.e., a node neither maintains global routing table nor exchanges it periodically), and relies on dynamically establishing routing table entries in the nodes.

1) *Path discovery process*: this process is performed before the source *S* really transmits data packets to destination *D*. The paths discovery starts by broadcasting a route request (*RREQ*) short message by *S* to its neighbors. As the request propagates from *S* to *D*, it carries information about the flow state serving to calculate the end-to-end delay, bandwidth, and hop count for such a given path.

The intermediate node performs the admission control policy (see Section IV.B) at soon as it receives *RREQ* packet (Fig. 3). If the *RREQ* is accepted, the node will add a route entry in its routing table with status *checked* and rebroadcast the request to the next hop. This status remains valid for a short period adapted dynamically according to the network resources state by the soft-states reservation mechanism as will be explained in Section IV.E. If no reply packet arrives at this node in time, the route entry will be deleted at the node and the reply packets coming after this time will be ignored. Each *RREQ* is identified by the (*source_addr*, *sequence*, *Broadcast_ID*). The fields *hop_req*, *del_req*, and *band_req* indicate respectively the required hop count, delay and bandwidth. These fields are useful for the admission control process based on hop count and /or end-to-end delay and/or available bandwidth. Note that the *RREQs* already processed are discarded by the node by keeping track of *RREQ* identifier. In order to limit the time and length of path discovery, in terms of hop count, the variable *time_to_live (TTL)* is decremented at each hop. If it counts down to zero, the *RREQ* is dropped and no further processed. This is because we consider that even if a long path which satisfies the QoS requirements (such as bandwidth) may exist, the dynamic topology of ad hoc networks makes difficult to maintain it.

<i>packet_type</i>	<i>source_addr</i>	<i>dest_addr</i>	<i>sequence</i>	<i>Broadcast_ID</i>
<i>previous_addr</i>	<i>TTL</i>	<i>hop_req</i>	<i>del_req</i>	<i>band_req</i>

Figure 3. RREQ format

<i>packet_type</i>	<i>source_addr</i>	<i>dest_addr</i>	<i>sequence</i>
<i>hop_req</i>	<i>del_req</i>	<i>band_req</i>	

Figure 4. RREP format

2) *Path reservation process*: the destination node initiates the reservation process upon receiving the first *RREQ* by generating one route replay (*RREP*) short message. *RREP* is then unicasted back to the source along the reverse route. An intermediate node which receives an *RREP* (Fig. 4), checks its resources availability and updates its route status to *reserved* if the packet is accepted. Note that any intermediate node may initiate the reservation process if it knows the route to the destination with corresponding sequence number greater than or equal to that contained in the *RREQ*, by appending information to the partial route into an *RREP*, and sending it back to the source. The fields *hop_rep*, *del_rep*, and *band_rep* corresponding to the QoS criteria are initialized at the beginning of reservation process, and updated when an intermediate node receives *RREP*. Hence, the source node will detain at the reception of *RREP*, a view about the network status which allows it to make a decision about transmission.

The routing scheme uses the sequence number to avoid possible loops during path discovery. The sequence number is also useful for detecting stale cached routes.

B. The policy of multi-criteria admission control

The policy of bandwidth control and end-to-end delay control is discussed in hereafter. The QoS control sub plan tries to find routes that satisfy a maximum of constraints between source *S* and destination *D*. For each constraint, a set of paths and their resource measurements (which represent the guaranty intervals) are calculated. Note that in our discussion about service class selection and guaranty intervals at a particular node, the word “path” designates the next hop in that node routing table.

1) *Bandwidth control*: the bandwidth control is very useful for the optimization of resources using. In our study, the value of the available bandwidth β at a node *i* is function of the node bandwidth capacity BC_i and the traffic T_i at this node.

We define the *bandwidth domain* $\varphi(i)$ of a given node *i* as the domain which covers the first and the second vicinities of *i*:

$$\varphi(i) = \{T_{ij}/\omega(i,j), \omega(\omega(i,\kappa),j)\}$$

where $\omega(i,j), \omega(\omega(i,\kappa),j)$ are the neighboring relations defined as:

$$\omega(i,j) = \{i \text{ has a link with } j\}$$

The bandwidth available β_i at node i is then given by (1):

$$\beta_i = BC_i - \sum_j T_{ij}, \quad \forall T_{ij} \in \varphi(i) \quad (1)$$

Therefore, the available bandwidth at node i depends on the traffic generated at this node, the traffic travelling through this node, and the traffic generated at the neighboring nodes. We use (1) to determine the path offering the maximum bandwidth among n paths. Assuming that $N_l \beta_k$ is the bandwidth at node k in path l , then (2) gives the minimum available bandwidth within each path.

$$X_b^{(n,1)} = \begin{cases} \text{MIN}(N_1 b_0 + \dots + N_1 b_k + \dots + N_1 b_m) \\ \text{MIN}(N_i b_0 + \dots + N_i b_k + \dots + N_i b_m) \\ \text{MIN}(N_n b_0 + \dots + N_n b_k + \dots + N_n b_m) \end{cases} \quad (2)$$

$$\delta_{-B^1} = \text{MAX}(X_b^{(n,1)}) \quad (3)$$

where δ_{-B^1} denotes the bandwidth of the path we are looking for. Let *band_req* be the value of the requested bandwidth in the *RREQ* message arriving at node k in path l . (3) is solved by using the following process. During the travelling of the *RREQ* along a path l , each node k calculates $\text{MIN}(\text{band_req}, N_l \beta_k)$ and makes a decision to forward *RREQ* if the bandwidth requested is available, otherwise the request is discarded. Upon receiving each *RREQ*, the destination will initialize randomly the field *band_rep* to a value higher than its bandwidth value. Then, a *RREP* is generated and sent back to the source. As soon as *RREP* is received by an intermediary node k on path l , the calculation of $\text{MIN}(\text{band_rep}, N_l \beta_k)$ is performed and stored in the node routing table and then set to a newly generated *RREP*.

Upon receiving the first *RREP* by the source node, a sorting algorithm for computing the maximum bandwidth is activated, and is performed at the reception of other route response packets. δ_{-B^1} is the maximum bandwidth value among the n *RREPs* received. P^1 points out the associated path.

Now we have to identify the guaranty interval service class (GISC). For that end, we extract the second feasible bandwidth δ_{-B^2} using (4).

$$\delta_{-B^2} = \text{MAX}(X_b^{(n,1)} - \delta_{-B^1}) \quad (4)$$

$GI(B^1) = [\delta_{-B^2}, \delta_{-B^1}]$ is the first guaranty interval for bandwidth constraint; the feasible path associated is noted P^1 . Similarly, we identify the second GISC:

$$\delta_{-B^3} = \text{MAX}(X_b^{(n,1)} - \delta_{-B^2}) \quad (5)$$

$GI(B^2) = [\delta_{-B^3}, \delta_{-B^2}]$ represents the second guaranty interval, with P^2 being the feasible path associated. We can calculate other guaranty intervals as follows:

$$\delta_{-B^k} = \text{MAX}(X_b^{(n,1)} - \delta_{-B^{k-1}}) \quad (6)$$

and P^k is the path generated by δ_{-B^k} , it will be the path associated to $GI(B^{k-1})$.

The utility of these guaranty intervals is described in Section IV.C.

2) *End-to-end Delay control*: in this control scheme, we search for the routes offering a minimum delay between S and D . Assume that $N_j d_j$ is the delay at the node j in the path i , which is the time required by j to receive and process packets as detailed in (7). Then, the minimum end-to-end delay between S and D is $\text{MIN}(X_d^{(n,1)})$, and the associated path will be the optimal path in terms of delay.

$$X_d^{(n,1)} = \begin{cases} \text{del}_{path} = N_1 d_0 + \dots + N_1 d_k + \dots + N_1 d_m \\ \text{del}_{pathi} = N_i d_0 + \dots + N_i d_k + \dots + N_i d_m \\ \text{del}_{pathn} = N_n d_0 + \dots + N_n d_k + \dots + N_n d_m \end{cases} \quad (7)$$

Our routing scheme does not try to find the minimum end-to-end delay only, but also other feasible delay values in order to build the intervals of service classes. Upon receiving a route request by node j and before sending it to node $j+1$, the computation $(\text{del_req} - N_j d_j)$ is performed, and stored in the node routing table. Then, it is set into a newly generated *RREQ* which will be sent to the next node (*del_req* is the requested delay). Note that this computation is useful and necessary to the node for making a decision about the satisfaction of delay requirement. If $(\text{del_req} - N_j d_j) < 0$, then the route request is discarded.

Once D receives a request, it initializes the path calculation where it sets *del_rep* = 0, and then $(\text{del_rep} + N_i d_k)$ is computed at each intermediate node k on path i back to S , stored in the node routing table, and sent into a new route replay. The minimum of path delay is determined at source node by using a sorting algorithm that we implemented in NNMS kernel. Among some n *del_rep* received, the selection is performed on the minimum delay, noted δ_{-D^1} , the second minimum delay δ_{-D^2} , the third δ_{-D^3} , etc. In (8) MIN^k gives a minimum of order k .

$$\delta_{-D^k} = \text{MIN}^k(X_d^{(n,1)}) \quad (8)$$

δ_{-D^1} is the minimum delay, having P^1 as the associated path.

The other minimums are determined as follows:

$$\delta_{-D^2} = \text{MIN}(X_d^{(n,1)} - \delta_{-D^1}) \quad (9)$$

$GI(D^1) = [\delta_{-D^1}, \delta_{-D^2}]$ defines the first guaranty interval for the delay constraint, and the path associated is P^1 .

Similarly, we identify other guaranty intervals:

$$\delta_{-D^k} = \text{MIN}(X_d^{(n,1)} - \delta_{-D^{k-1}}) \quad (10)$$

where the path generated by δ_{-D}^k, P^k , is associated with the $GI(D^{k-1})$ defined by (11):

$$GI(D^{k-1}) = [\delta_{-D}^{k-1}, \delta_{-D}^k] \quad (11)$$

3) *Hop count control*: the calculation of the feasible paths based on hop count and their corresponding guaranty intervals is similar to the calculation of the end-to-end delay, because the hop count control is also an additive metric that needs to be minimized.

C. Guaranty intervals

The guaranty intervals service classes (GISC) are used by the source node to select the ad hoc (adequate) paths for data transmission. GISC are composed of the interval values and their associated feasible paths. The algorithm presented in Fig. 5 permits to check and select the suitable service class for a flow bandwidth requirement (similar algorithms have been developed for other QoS constraints). If the QoS requirement is within the GISC interval, then this class is considered to be suitable and also to be the best. Otherwise, if the flow requirement is less than the lower value of the interval of some available service classes, then all these classes are considered to be suitable, and the protocol selects a service class from the suitable ones based on some criteria. The criteria can be for example, a) the suitable class with the biggest lower interval value; b) the suitable class with the smallest lower interval value; or c) simply a random suitable class. Note that similar processing is performed for other QoS constraints.

Theorem: Let consider A_1, A_2, \dots, A_n be the paths relying source node S to destination node D , and $f_m(A_1), f_m(A_2), \dots, f_m(A_n)$ the associated measurement functions in terms of QoS metric m . Then, the problem of determining the path which offers a better metric performance within guaranty interval $[\alpha, \beta]$ is a resolvable problem.

Proof: we say that A_j has better performance than A_i for the metric m with parameter s if: $f_m(A_i) < f_m(A_j)$: $\int_{\alpha}^{\beta} [f_m(A_i(s))]' ds / (\beta - \alpha)$ is the derivate of $f_m(A_i)$ over the interval $[\alpha, \beta]$.

Then, the comparison of QoS satisfaction of A_i and A_j over $[\alpha, \beta]$ is given by the equation (12):

$$\int_{\alpha}^{\beta} (f_m(A_i(s)) - f_m(A_j(s)))' ds / (\beta - \alpha) = (f_m(A_i(\beta)) - f_m(A_j(\beta)) - (f_m(A_i(\alpha)) - f_m(A_j(\alpha)))) / (\beta - \alpha) \quad (12)$$

A_j offers better performance than A_i if:

$$f_m(A_i(\beta)) - f_m(A_i(\alpha)) < f_m(A_j(\beta)) - f_m(A_j(\alpha))$$

Let $\Omega(A_i) = f_m(A_i(\beta)) - f_m(A_i(\alpha))$

$$\Omega(A_j) = f_m(A_j(\beta)) - f_m(A_j(\alpha))$$

```

Check availability ( $B\_flow$ );
{
  Selected = false /*initialization of a variable indicating
  the result of selection*/

  /*we traverse service classes (the number of class is
  U)*/
  While ( $CS < U$ )
  {
    If ( $B\_flow \in ] \beta_{cs}, \beta_{cs+1}[$ )
      /* the convenient service class to  $B\_flow$  is reached */
       $PBCs = PTSELECT (] \beta_{cs}, \beta_{cs+1}[)$ 
      /* activate the feasible path  $PBCs$  corresponding */
       $ACTV (PBCs)$ 
      Break;
    }
  }
  /* the bandwidth requirement does not found a suitable
  service class */
  If selected = false then
  {
    /*select the smallest service class  $CS$  where  $B\_flow$  is
    less than  $\beta_{cs}$  */
    If exist  $CS$  where ( $B\_flow < \beta_{cs}$ )
    {Select MIN ( $CS$ )
      $PBCs = PTSELECT (] \beta_{cs}, \beta_{cs+1}[)$ 
     Selected = true
    }
  }
}
/* Start the paths discovery if there is no suitable service
class*/
If selected = false then Start ( $Paths\ discovery\ process$ )
}
    
```

Figure 5. Algorithm for GISC bandwidth constraint

Then,

A_i has better performance than A_j if: $\Omega(A_i) > \Omega(A_j)$

A_i has less performance than A_j if: $\Omega(A_i) < \Omega(A_j)$

A_i has the same performance as A_j if: $\Omega(A_i) = \Omega(A_j)$

D. Priority Queuing

The prioritization scheme is one of the tools used to achieve a service differentiation in wireless ad hoc networks. SWAN model uses the priority queuing tool by limiting the amount of real-time traffic in order to process the lower priority packets. FQMM model uses a simple priority queue to ensure that high-priority packets are processed before low-priority ones. Our model implements a separate queue for each class. The GISC shares a buffer space; when buffer is not occupied by one service class, it is assigned to other classes if there is a need. By using priority queuing, the congestion prediction and prevention becomes possible at node level: before occupying all buffer space, the node informs its neighbors to slow down the traffic when sending ACK packet (the reception of ACK packet indicates that a data packet was received successfully).

The rate decrease (*RD*) of traffic is calculated according to the priority queuing state (small or large congestion) of the node having sending ACK. Consequently, the new traffic rate (*NT*) of transmission would be:
 $NT = OT - (OT * RD) / 100$,

where *OT* is the old traffic rate. Hence, the traffic is adapted according to the network state.

E. Detection and Recovery of QoS violation

In ad hoc networks, the mechanisms permitting the detection and recovery of QoS violation are of central importance in order to support multimedia applications.

1) *QoS violation detection*: To address this issue, we propose two methods: the first one is based on different states of resource reservation on intermediate node, as for second one, it is based on the time of receiving data packets. These two methods are explained hereafter.

a. *Soft-states resources reservation for QoS violation detection*: The resources available in an intermediate node may take one of four states (Section IV.A): *checked*, *reserved*, *allocated* (at the reception of data packet), and *liberated* (once the resources are not used). Each state has a short interval called “soft-state interval” (Fig. 6). During the routing process, the resources available are temporarily reserved to the flow for a short time δ . When an intermediate node receives a data packet it allocates the resources reserved over the next interval. Therefore, if a new packet is not received within the soft-state interval, then resources are released and flow states are removed in a fully decentralized way.

Let consider:

Δt the soft-states interval (time difference between two states *i* and *j*) of an intermediate node.

μ the target satisfaction which defines the desired percentage of packets to be sent within the QoS constraint, where $\mu = 1$ corresponds to the best QoS guaranty and $\mu = 0$ corresponds to the best-effort transmission.

Then, (13) verifies the probability that Δt is less than the time requested value δ and the session request to be accepted.

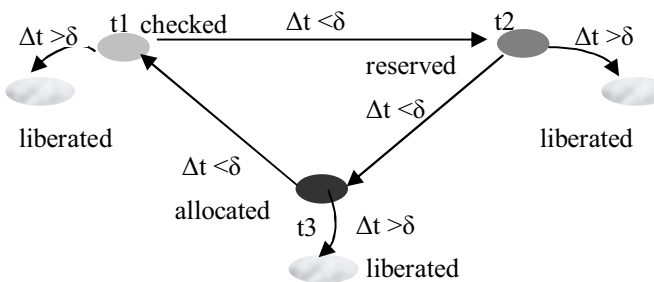


Fig. 6. Soft-states of an intermediate node NNMS

$$P[\Delta t \leq \delta] \geq \mu \tag{13}$$

The exact evaluation of (13) requires the destination to be acquainted with the statistical descriptions of delay of each node along the path. However, in many cases, the statistical distribution of such parameter can be approximated by a Gaussian distribution. Under this hypothesis, and assuming independency among nodes statistics, the soft-states interval turns out to be a Gaussian variable. Let m_{Sp} and σ_{Sp}^2 be the statistical average and the variance of the random variable S_p , respectively (S_p is the soft-states time of a given node). Therefore, the soft-states interval statistics can be expressed as follows:

$$P[\Delta t \leq \delta] = 1 - Q\left(\frac{\delta - m_{Sp}}{\sigma_{Sp}}\right) \tag{14}$$

Where Q represents the complementary distribution function of a Gaussian variable with mean 0 and variance 1.

Let consider δ and μ be the maximum time and satisfaction target for an accepted session request. Let ν be the actual time satisfaction provided by the intermediate node as given by (14). Hence, a QoS request flow would be satisfied even the average soft-state interval was increased to the value m_{Sp} given by:

$$m_{Sp} = \delta - \sigma_{Sp} Q^{-1}[1 - \nu] \tag{15}$$

As a result, the QoS violation is detected at an intermediate node if the soft-state time is greater than m_{Sp} (m_{Sp} is the time bound of the soft-state interval).

This tool of QoS violation detection is also useful for optimizing the use of available bandwidth because it permits to liberate the reserved bandwidth if it is not allocated to any flow.

b. *Delay QoS violation detection*: we use this mechanism to detect the QoS violation during the transmission of data packets. It triggers automatically the recovery process (Section IV.E.2) when the time estimated to receive a packet is exceeded. This excess is caused either by broken links or congestion as a result of traffic augmentation in the network. Let denote:

T the time needed to generate an *RREQ* at node *i*, which has del_node as the current delay available at its routing table.

ΔT_T the estimated packet processing time at source node.

ΔT_G the laps of time between sending two consecutive data packets by the application generating the traffic.

Now, we can compute the estimated time *TEXC* for receiving data packet *j* by node *i*.

$$TEXC(j) = T_i + \Delta T_T + \sum^{j-1} \Delta T_G \tag{16}$$

Thus, the QoS violation can be easily detected at a given node by monitoring the delay of the arriving data packets as expressed in (16). If a node receives data packet *j* whose delay exceeds the maximum time

requirement $TEXC(j)$, a QoS violation by time exceeding is detected and the QoS recovery mechanism will be triggered.

2) *QoS violation recovery*: the priority queuing is an efficient way to prevent QoS violation caused by the network congestion. This is achieved by decreasing the traffic rate when an excess traffic rate is detected. On the other hand, NNMS applies a recovery mechanism that can be used to re-establish the route along a new path when a path is broken. The approach adapts the routing paths according to the new network state caused either by nodes mobility or degraded path state. When a QoS violation is detected at a node x , a neighbor node z will send a short update message to the destination D . Then, D will broadcast back to the source S an update message. The same computations as in the route discovery process in terms of admission control will be performed during the traveling of the update message to S . Note that loops are avoided by decreasing the sequence number value of the node z before it initiates the recovery process.

After having described the different functions of the kernel, we will present hereafter the applied multilayer feedforward neural networks to NNMS kernel.

V. INTELLIGENT LEARNING PLAN

The intelligent learning plan performs the learning of the NNMS kernel operations. For that end, we use neural networks. The neural networks tool is a robust artificial intelligence technology that can handle a complex and dynamic nature of various processes. As such, this technology has been gradually gaining acceptance in the industry treatment as a tool for process modeling and control.

One main functional characteristic of neural networks is its distributed capability because of its particularly parallel architecture. It possesses many inputs and treats the problem in parallel way. Using of appropriate form of training data makes processing time independent of the traffic load in the network. Furthermore, neural networks are also useful for optimizing the decision rules adaptively based on the changes of the environment and the states of the system [17]. In our study, the multilayer feedforward neural network is used for optimizing the NNMS kernel decision adaptively according to the network state.

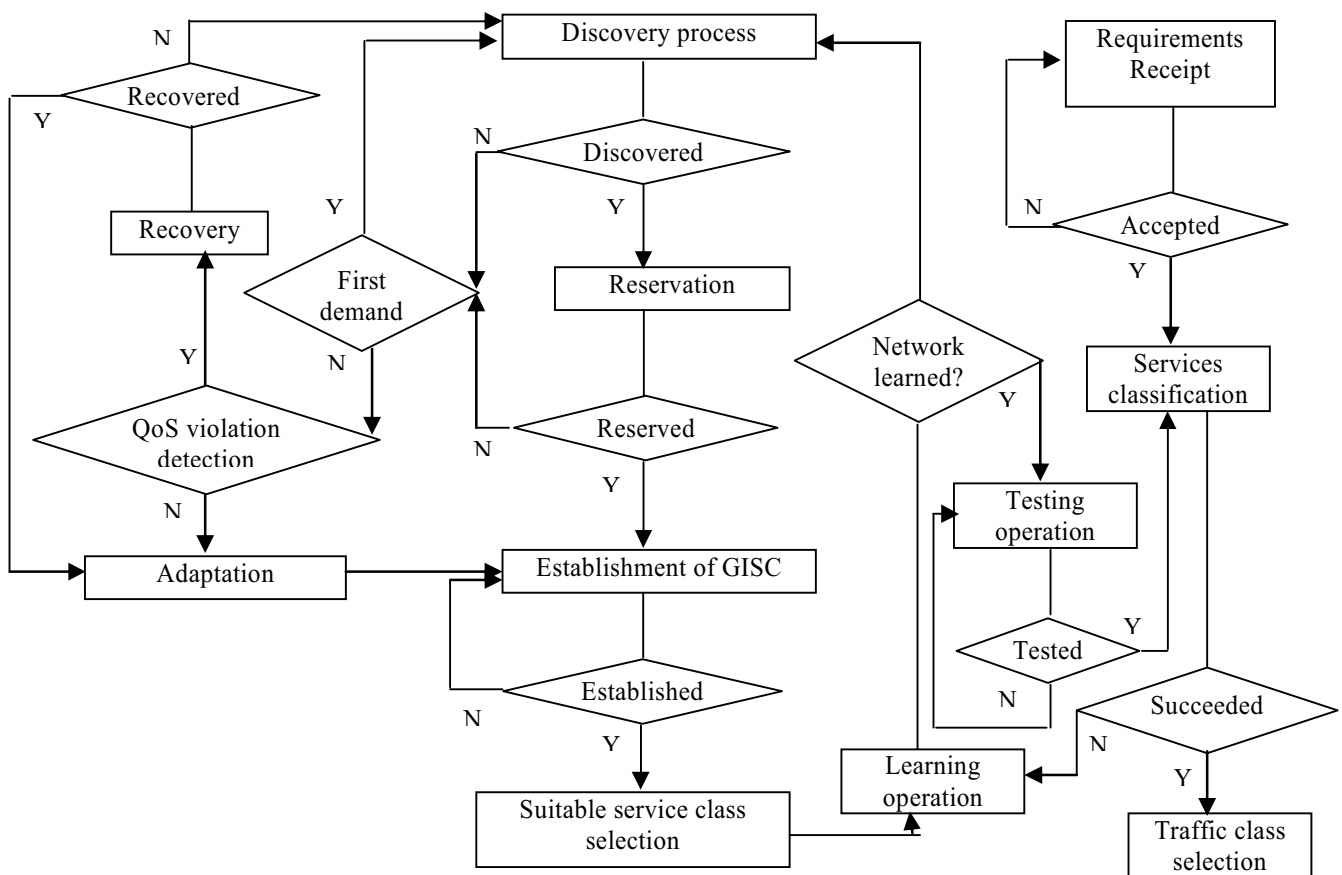


Figure 7. Schematic diagram of NNMS operations

A. Multilayered Feedforward Neural Network (MFNN) and its application to NNMS

MFNN is considered as one of the best neural network models used in practice because of its simple architecture and various domains of applications. The processing components of MFNN are called *neurons*. The neurons are interconnected together by a modifiable means named *weights* representing the strength of link weight [17]. The network is divided into layers: input layer consists of the inputs to the network; hidden layer consists of number of neurons placed in parallel where each neuron performs a weighted summation of the inputs, and then generates via transfer function an output through output layer. In order to assure the adaptive behaviour of the network, the correlation between successive levels of neurons is changed when the weights values are modified. The adaptive behaviour process is known as *training*. The training is the process of estimating the connecting weights by minimizing some overall error measures, using mechanisms such as Back Propagation algorithm (BP). BP is widely used neural network training algorithm. Mathematical details related to BP are described in many neural networks references such as [17] and [18].

In order to train a neural network to perform some task, the weights of each unit should be adjusted in such a way that the error between the desired output and the actual output is reduced. This process requires that the neural network compute the error derivative of the weights. The back propagation algorithm is widely used method for determining this error named MSE (Mean Squared Error). It is calculated as the mean square difference between the desired output and the actual output. Since we look for minimizing the error for each weight separately, the overall error may increase. Then, we should calculate the total output error after each adaptation when that error is greater than the previous rejected error, after that we calculate new learning rates. The modification of weights continues until the actual output approximates the desired response. The long convergence time problem of BP is solved by incorporating the weight momentum term in BP.

B. Application of MFNN to NNMS

A schematic diagram of NNMS operations is presented in Fig. 7. It shows the relationship between the NNMS kernel operations and their learning process using neural networks. The operations (i.e., the functionalities described in Fig.1 and Fig.2 such as admission control, etc.) are represented by squares, and the check of success of operations is represented by rhombus ('Y' and 'N' signify respectively the success and the failure of the operation).

The application MFNN over GOQS is divided into two phases. The first phase is the training of the neural

network with the training data set and the testing of the one after the training. The second phase is the prediction and selection of service class and next hop to forward the data packets. This is solved by the next algorithm:

1. The creation of the neural network.
2. The division of the training data into sets at random (the number of the one data set is 1000).
3. The training of neural network with each data set and in result we receive the prediction output.
4. The testing each prediction output by whole set of experimental data and the selection of the output, which has the maximum level of correctness.
5. Prediction of the potential service class and next hop to forward the data packets from testing data set by the best output.

The steps 1-4 consist of the first phase; the step 5 is the phase stage.

Once the network is learned (steps 1-4) with the data offered by the kernel mechanisms, it would be possible to select the suitable service and next hop (step 5) in order to satisfy the flow requirements. The benefit of this technique as will be shown in the simulation is the reduction of the time required by the different mechanisms of NNMS kernel.

As shown in Fig. 7, the network training stage is realized after several scenarios of execution of the kernel operations such as discovery process, reservation process, traffic adaptation via queuing priority, QoS violation detection and recovery, etc. Once the training stage of the kernel is achieved, NNMS starts the testing operation over additional scenarios in order to test the correctness of the results of the first stage. The success of the network learning would be able to satisfy the flow requirements without performing the operations required by the kernel.

We have chosen a three-layer MFNN model. For training the neural network we have used the resilient back propagation training algorithm. The performance function is mean square error, which has been used to calculate the differences between the target output and the network output. For the neurons of the input and hidden layers we have used the tan-sigmoid transfer function. The neuron of the last layer has the linear transfer function. The dimension of the input patterns that represent the QoS requirements (delay, bandwidth, and hop count required by the user) specifies the number of input nodes. Concerning the number of hidden layer nodes, according to the Kolmogorov theory [19], if the number of input nodes is N , then $2N+1$ hidden node can be used. The output nodes are determined by the number of categories to be classified. In our case, we classify three categories which correspond to the traffic with delay constraint, traffic with bandwidth constraint, and best-effort traffic. For each category, MFNN gives the next hop for routing the traffic received. We opt for a

binary classification, then each bit of a desired output presents either state 1 or state 0: state 1 signifies “belong to” and state 0 signifies “does not belong to” a given category (i.e., 001 means that the desired output pattern belong to category 1).

VI. SIMULATION

The performance of the presented QoS model is studied with simulations. The main functionalities of NNMS kernel is implemented with GlomoSim [20]. We investigated NNMS performances with different node speeds and multiple traffic flows. The simulation environment consists of 50 nodes moving in an area of 1000 by 1000 m. The transmission range of nodes is 200 m, while their transmission rate is 2 Mb/s. The random way point mobility model is implemented at each node in the network. In the beginning, the nodes are randomly placed in the area. Then, each mobile node selects a random destination and moves with a random speed up to a maximum speed of 20m/s. After reaching the destination, the node will stay there for a given “pause time” then starts to move towards another destination. This process is repeated during all simulation time.

The simulation is realized in two steps: a) the first step investigates the performances of NNMS kernel compared with both the ‘original model’ and SWAN model described in [1]. We use the word “original model” to refer to wireless ad hoc networks without the NNMS kernel mechanisms. The original model is simulated by using the IEEE 802.11 MAC protocol with the Dynamic Source Routing (DSR) [21], b) in the second step, we

exploit the data generated during the first step in order to study the performances of the training process of the intelligent learning plan. The simulations include a mixture of traffic types; TCP flows, voice flows and video flows.

Both best-effort traffic (TCP flows) and real-time traffic (voice and video flows) are active and monitored for the duration of 600 s. The source and destination nodes associated with flows are distributed among the mobile nodes in the wireless ad hoc network. TCP flows are greedy FTP type of traffic with packet size of 512 bytes. Voice traffic is modeled as 32kbps constant rate traffic. Video traffic is modeled as 200kbps constant rate traffic with a packet size of 512 bytes.

Figs. 8 and 9 show respectively, the average end-to-end delay and the average throughput of traffic with various values of mobility. It is observed in Fig. 8 that the average end-to-end delay in NNMS is reduced by 40-60% compared to the original model (IEEE 802.11). Furthermore, the average throughput of NNMS as illustrated in Fig 9, decreases slowly from 99% to 91%, whereas the average throughput in the original model decreases from 76 % to 54%. For different scenarios of mobility in our model, the average end-to-end delay grows very slowly, and the traffic throughput becomes almost constant with less than 9 % reduction in throughput. These results demonstrate that we can achieve a reduction of 40-60% in the average end-to-end delay of traffic with 9% loss of traffic throughput using NNMS kernel mechanisms. The interpretation of these promising results of NNMS is discussed hereafter where we compare our model with SWAN model.

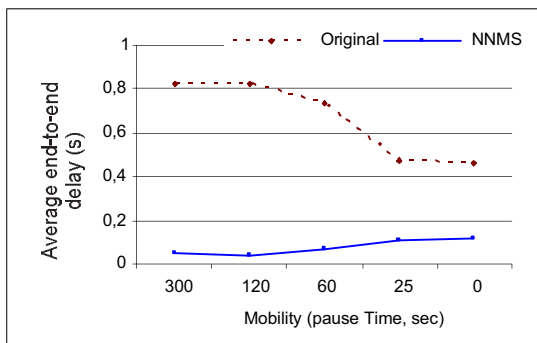


Figure 8. Average end-to-end delay versus mobility

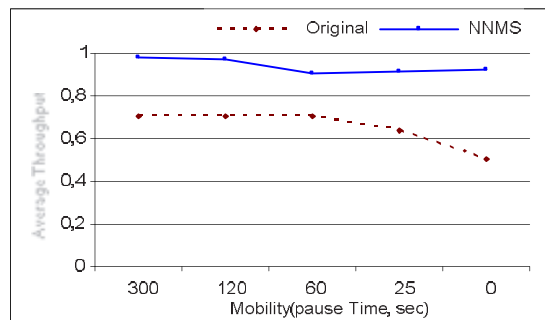


Figure 9. Average throughput versus mobility

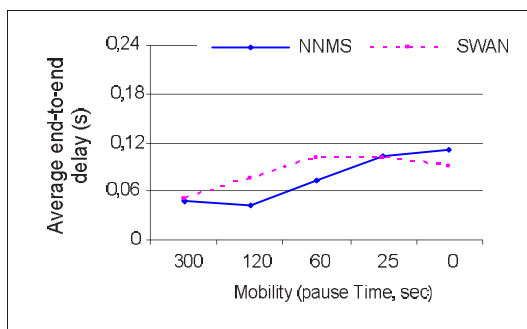


Figure 10. Average end-to-end delay versus mobility

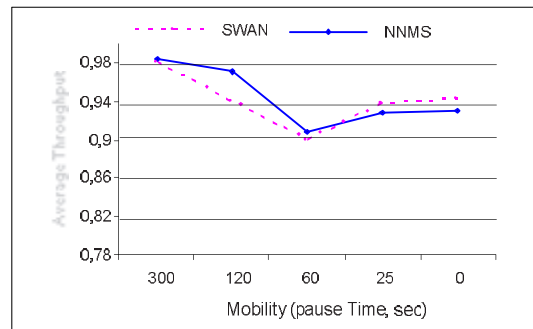


Figure 11. Average throughput versus mobility

The impact of mobility on the average end-to-end delay in NNMS and SWAN is illustrated in Fig. 10. It is observed that the average end-to-end delay in NNMS increases slowly as the mobility increases, and it grows only for the highest mobility scenarios. The average delay offered by NNMS is about 10 % better than that offered by SWAN at non-frequent network mobility. This shows the efficiency of the implemented kernel mechanisms and the intelligent learning plan.

The congestion level is efficiently regulated with the suitable decrease and increase traffic rate using priority queuing. In addition, critical situations such as route interruptions are well managed by the detection and recovery mechanisms. However, for the highest mobility scenarios, the average end-to-end delay in SWAN is slightly smaller than that of NNMS. The impact of high mobility on delay is due essentially to the latency of the route discovery and the route interruption. In our model, the impact is due also to an additional computation time of guaranty intervals (which offer a suitable service according to the flow requirement), soft-states reservation and estimation of data receipt performed by the detection and recovery of QoS violation mechanisms. In the case of SWAN, its stateless nature makes it more resistant to mobility factor. At high nodes mobility, SWAN acts better by using the rate control of TCP and UDP traffic, and it ensures that if the delay increase beyond a certain threshold, some flows will be dropped in order to maintain low delay for the remaining flows. The gain of NNMS under non frequent mobility is very useful for supporting multimedia applications with delay constraint in non-highly dynamic network.

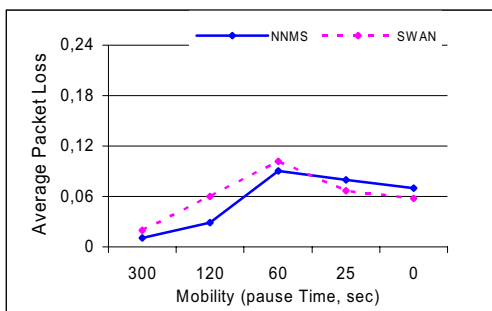


Figure 12. Average packet loss versus mobility

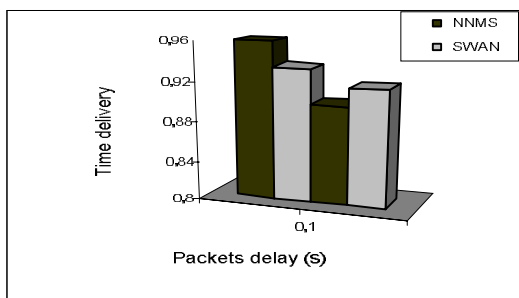


Figure 13. Distribution of the delivery packets

Fig. 11 shows the average throughput versus node mobility of both NNMS and SWAN models. We observe that for lower and medium mobility, the average throughput in NNMS decreases slowly from 99% to 91%, whereas SWAN average throughput decreases from 98% to 88%. As shown in Fig. 12, the packet loss of SWAN increases more than that of NNMS as mobility increases. This shows the efficiency of mechanisms we have used for detection and recovery of QoS violation. Note that the target satisfaction parameter of the soft-states resource reservation is set to 1 which means that we look for best QoS guarantee.

At lower and medium mobility, the estimation precision of data receipt and the soft-states reservation at intermediate node are efficiently performed, reducing then the packets loss by about 8% compared to SWAN. This latter relies only on feedback delay received from the MAC layer as a measure of congestion in the network. Nevertheless, other factors such node speed are essential in determining the accurate resources available. This explains why some flows were being dropped. On the other side, it is observed that for higher mobility scenarios, SWAN slightly outperforms our model. The impact of mobility on throughput is due to the broken links in the route relaying source and destination and to the increase of collisions, which may generate error messages and cause congestion along the new route. As the mobility increases in NNMS, the accuracy of data receipt estimation done by the delay QoS violation mechanism is reduced and the node states are frequently changed complicating then the calculation done by the soft-states reservation mechanism. At highest mobility scenarios, some flows are dropped in both SWAN and NNMS because of the difficulty in capturing the dynamics of the environment in ad hoc network. The average throughput of traffic in NNMS still more than 90% even for higher mobility scenarios.

A simple distribution of the delivery packets of both NNMS and SWAN is described in Fig. 13. It is shown that NNMS has more packets with packet delay smaller than 0.1s. About 96.5 % of packets with delay less than

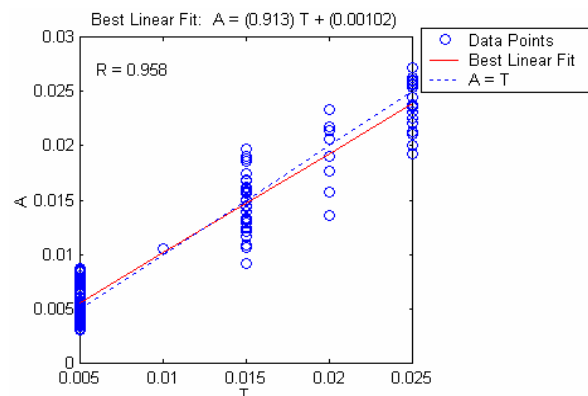


Figure 14. Network outputs plotted versus the targets

0.1s are received by the destination against 93% of SWAN packets. This result is useful for real-time services such as VOIP where the time constraint is very important. The cooperation between different mechanisms and plans of NNMS has provided the result above. We observe also that 89% of the packets having delays of more than 0.1s in NNMS, in comparison to 92% for SWAN, are received by the destination. We remark that our model outperforms SWAN by about 10% at lower and medium mobility in terms of average delay and throughput.

In what follows, we focus on the intelligent learning plan. For that end, we exploit the information provided by NNMS kernel. Using neural networks, we have designed an instrument that can identify the suitable service class and the QoS satisfaction level from resources measurements given by NNMS kernel in order to minimize the time of transmission. We have generated a total of more than 1000 data for which we have measurements of different QoS requirements. We divided the data up into training, validation and test subsets.

The neural network algorithm implemented contains three-layers as we have explained in Section V.B, with tan-sigmoid transfer function in the hidden layer and a linear transfer function in the output layer. We classify three outputs categories which correspond to the traffic with delay constraint, traffic with bandwidth constraint, and best-effort traffic. The Levenberg-Marquardt algorithm [22] is selected to assure the training function because of its better convergence properties [18] than the conventional backpropagation algorithms. Once the training is realized, the source node should be able to select the next hop for routing the data packets without performing any other treatment such as the route discovery. The next step is to perform some analysis of the network response. We put the entire data set through the network and perform a linear regression between the network outputs and the corresponding targets (Fig. 14).

The graphical outputs resulted by the training function is illustrated in Fig. 14. The network outputs are plotted versus the targets as open circles. A dashed line indicates the best linear fit. The perfect fit (outputs equal to targets) is indicated by the solid line. The values of the two parameters of Best Linear fit A (0.913 and 0.00102) correspond respectively to the slope and the y-intercept of the best linear regression relating targets to network outputs. We can see that our A values are close to the suitable case (1 and 0 respectively). The R -value that represents the correlation coefficient between the outputs and targets is equal to 0.958 (a perfect correlation is reached when R is equal to 1). R measures how well the variation in the outputs is explained by the targets, the value found in our case reflects this best variation in the NNMS intelligent leaning plan.

Figs. 15, 16 and 17 show respectively, the ratios 0.1, 0.5, and 0.8 of the learning rate of the Levenberg-Marquardt training algorithm. We observe that the squared error equals about 10^{-5} after about 10 epochs of the training process in the three cases, which means a good traffic class satisfaction to the source requirement. The graphs shows the duration of the training required for selecting a next hop to perform the data delivery.

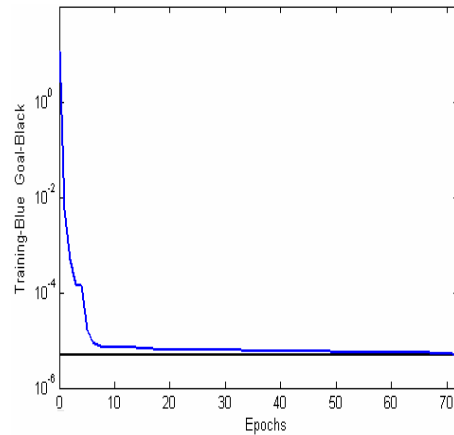


Figure 15. Squared Error of NNMS traffic classification with incrementing rate = 0.1

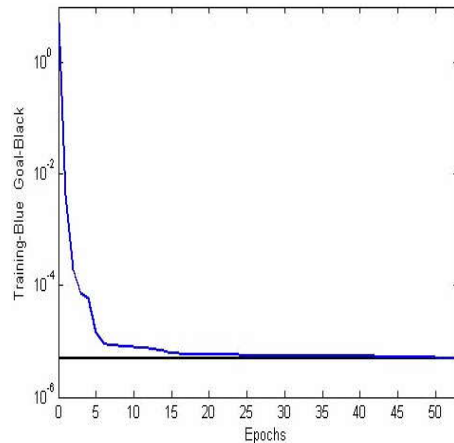


Figure 16. Squared Error of NNMS traffic classification with incrementing rate = 0.5

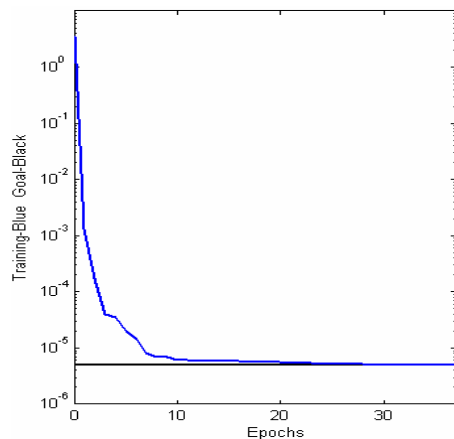


Figure 17. Squared Error of NNMS traffic classification with incrementing rate = 0.8

The multilayered feedforward neural network, as shown in Figs. 15, 16 and 17, is able to classify all the traffic according to the QoS requirements and to deliver data after 10 epochs. In other words, if we consider that each epoch corresponds to a number of cycles of delay delivery and the gain obtained is inversely proportional to the number of epochs, then the end-to-end delay required to transmit data packets from source to destination is minimized by about 10%.

The gain observed from the graphs is explained by the fact that the network learning has permitted a fast identification of the next hop for routing the data packets received at each node. Hence, there is no need to activate the process of path discovery. On the other side, for each input QoS requirements (delay, bandwidth, and hop count required) received by the source node, the network learning has performed a good traffic classification into outputs categories which correspond to the traffic with delay constraint, traffic with bandwidth constraint, and best-effort traffic.

VII. CONCLUSION

In this paper, we proposed a new QoS architecture for service differentiation based on neural networks in wireless ad hoc networks named NNMS. We presented the main operations and mechanisms of NNMS plans. The first plan named kernel plan, accomplishes the tasks of data routing and QoS support for the satisfaction of user requirements. Additionally, the kernel plan performs the traffic adaptation (using priority queuing) and the detection and recovery of QoS violation (using two complementary techniques we developed). The second plan named intelligent learning plan, completes essentially the learning of the kernel operations by means of multilayered feedforward neural networks (MFNN). The objective of using MFNN is the reduction of the nodes processing time, and as a result the delivery time of data packets.

The simulation results have shown the benefits of our model. NNMS provides performances of about 40% better than the original model (i.e. IEEE 802.11) and 10% better than SWAN in terms of end-to-end delay in networks whose topologies change at low to medium rate. Furthermore, an additional gain (about 10%) in terms of delay is obtained via the learning process accomplished by neural networks. These results illustrate that neural networks is very promising tool to support multimedia services.

ACKNOWLEDGEMENT

This work has been performed with the support of the National Sciences and Engineering Research Council of Canada.

REFERENCES

- [1] G.H. Ahn, A. T. Campbell, A. Veres, and L. H. Sun, "SWAN: Service Differentiation in Stateless Wireless Ad Hoc Networks", *IEEE INFOCOM* 2002.
- [2] S.-B. Lee, G.-S. Ahn, X. Zhang, and A.T. Campbell, "INSIGNIA: An IP-Based Quality of Service Framework for Mobile Ad Hoc Networks," *Journal of Parallel and Distributed Computing (Academic Press), special issue on wireless and mobile computing and communications*, vol. 60, no. 4, pp. 374-406, Apr. 2000.
- [3] H. Xiao, W. K.G. Seah, A. Lo, and K. Chaing "Flexible QoS Model for Mobile Ad-hoc Networks", *IEEE Vehicular Technology Conference*, Vol. 1, pp 445-449, Tokyo, May 2000.
- [4] G. Alonso J. Xue and P. Stuedi, "ASAP: An adaptive QoS protocol for mobile ad hoc networks", *IEEE PIMRC 2003*.
- [5] Y. L. Morgan, T. Kunz, "PYLON: An architectural framework for ad-hoc QoS interconnectivity with access domains", *HICSS'03 pres*, Hawaii, USA, January 2003.
- [6] L. Khoukhi, S. Cherkaoui, "Flexible QoS Routing Protocol for Mobile Ad Hoc Networks", *In Proc. of IEEE Int. Conference on Telecommunication (ICT2004)*, Brazil, Aug. 2004.
- [7] C. R. Lin and J.-S. Liu. "QoS Routing in Ad Hoc Wireless Networks", *IEEE Journal on Selected Areas in Communications*, Vol. 17, No. 8, 1426-1438, 1999.
- [8] Qi Xue and Aura Ganz. "Ad hoc QoS on-demand routing (AQOR) in mobile ad hoc networks", *Journal of Parallel and Distributed, Computing Elsevier Science*, USA, 2003.
- [9] A. Munaretto, H. Badis, K. Al Agha and G. Pujolle, "A Link-state QoS Routing Protocol for Ad Hoc Networks", *In Proc. of IEEE MWCN02*, Stockholm, Sept.2002.
- [10] S. Chen and K. Nahrstedt. "Distributed Quality-of-Service in Ad Hoc Networks", *IEEE Journal on Selected Areas in Communications*, Vol.17, No. 8, Aug., 1999.
- [11] W.-H. Liao, Y.-C. Tseng, J.-P. Sheu, and S.-L. Wang. "A Multi-Path QoS Routing Protocol in a Wireless Mobile Ad Hoc Network", *In Proc. of IEEE ICN'01: Conference on Networking, Part II*, pp. 158-167, July, 2001.
- [12] R. Sivakumar, et al., "CEDAR: a core extraction distributed ad hoc routing algorithm", *IEEE Journal on Selected Areas in Communications*, vol. 17, no. 8, pp. 1454-1465, 1999.
- [13] S. Chen and K. Nahrstedt, "On finding multi-constrained paths", *IEEE International conference on communication*, 874 -879 June 98.
- [14] Z. Wang and J. Crowcroft, "QoS routing for supporting resource reservation", *IEEE Journal on Selected Areas in Communications*, Vol. 14, No. 7, 1996.
- [15] C.-R. Lin. "On-Demand QoS Routing in Multihop Mobile Networks". *In Proc. of IEEE INFOCOM*, pp. 1735-1744, April 2001.
- [16] Y. Chen, Y. Tseng, J. Sheu, and P. Kuo, "On-Demand, Link-State, Multi-Path QoS Routing in a Wireless Mobile Ad-Hoc Network" *European wireless*, Florence, Feb. 2002.

- [17] J.Zurada, Introduction to Artificial Neural Systems. St. Paul, MN: West Publishing, 1992.
- [18] A. K. Jain, J. Mao, and K. M. Mohiuddin. "Artificial neural networks: A tutorial". *IEEE Computer*, March 1996, 29 (3) pp.56-63.
- [19] V. Kurkova. Kolmogorov's theorem. In Michael A. Arbib, editor, *The Handbook of Brain Theory and Neural Networks*, pages 501--502. MIT Press, Cambridge, Massachusetts, 1995.
- [20] Global Mobile Information Systems Simulation Library, GLOMOSIM, <http://pcl.cs.ucla.edu/projects/glomosim/>.
- [21] D. Johnson and D. Maltz. "Dynamic source routing in ad-hoc wireless networks", *In Computer Communications Review - SIGCOMM '96*, Aug.1996.
- [22] M. T., Hagan, and M. Menhaj, "Training feedforward networks with the Marquardt algorithm", *IEEE Transactions on Neural Networks*, vol. 5, no. 6, pp. 989-993, 1994.



Dr. Lyes Khoukhi is a Post doctoral researcher at the University of Montreal (Canada) since 2008. He received Ph.D degree in Electrical and Computer Engineering from the University of Sherbrooke (Canada) in 2007, and M.Sc degree in Computer Science from University of Versailles (France) in 2002. During 2003-2007, he stayed in INTERLAB

Communications Research Laboratory, Sherbrooke University. His research interests include wireless communications, mobile ad hoc networking, intelligent and distributed systems.



Pr. Soumaya Cherkaoui is an Associate Professor at Sherbrooke University, Canada which she joined in 1999. Since 2000, she is also the director of INTERLAB, a research group comprising more than 12 faculty and research assistants which conducts research funded both by government and industry. Before joining Sherbrooke University as a faculty

member, Professor Cherkaoui worked for industry as a project leader. She has over 50 publications in the areas of network protocols and distributed systems.