# Predictive Delay Metric for OLSR Using Neural Networks

Zhihao Guo
Case Western Reserve University
10900 Euclid Avenue
Cleveland, Ohio
01-216-3684114
zxg8@case.edu

Behnam Malakooti
Case Western Reserve University
10900 Euclid Avenue
Cleveland, Ohio
01-216-3684462
bxm4@case.edu

## ABSTRACT
In this paper, we propose an adaptability enhancement mechanism to be integrated with OLSR, and potentially any Mobile Ad Hoc Network (MANET) proactive routing protocol. The key of this mechanism is prediction and evaluation of the mean queuing delay as a routing metric. Neural network methods are used to predict delays. We investigated the pros and cons of using two types of neural networks, namely Multi-Layer Perceptron (MLP) and Radial Basis Function (RBF), in predicting nonstationary time series (e.g., mean queuing delay time series). We present TierUp – our novel node-state routing table calculation algorithm, which is developed for the integration of delay prediction and OLSR. We name the extended version of OLSR as OLSR_NN. We show through ns2 simulation that compared to OLSR, OLSR_NN is able to increase data packet delivery ratio and reduce average end-to-end delay in scenarios with complex traffic patterns and various node mobility. Our simulation also shows the advantage of using neural network for delay prediction compared to moving average and exponential smoothing. The enhanced adaptability of OLSR_NN is further verified by the more balanced traffic observed in our simulation.

## Categories and Subject Descriptors
C.2.2 [**Computer-Communication Networks**]: Network Protocols – *routing protocols, protocol verification.*

## General Terms
Algorithms, Measurement, Performance, Design, Verification.

## Keywords
Delay prediction, MANET proactive routing, neural network, TierUp, OLSR_NN

## 1. INTRODUCTION
One problem of the prototypes of existing typical MANET routing protocols is heavy dependence on hop count evaluation. Proactive protocols such as OLSR [10] and TBRPF [18] only evaluate hop counts to figure out the shortest path. As for reactive protocols, AODV [19] only checks hop counts to determine if the routing table entry should be updated; DSR [11] relies on hop counts for automatic route shorting and preventing route reply storms. Although it is cost efficient to solely evaluate hop counts in routing decision making, some valuable dynamic network conditions such as per-node delay and varying link stability are neglected. Over time those short cuts in terms of hop counts may be overused and loss their packet delivery efficiency. Thus efficient evaluation of routing metrics other than hop counts is potentially beneficial to enhance the adaptability of a MANET routing protocol.

Manuscript submitted to the Third Annual International Wireless Internet Conference.

In this paper, we focus on investigating this adaptability issue with respect to proactive MANET routing protocols. Specifically, our system is built as an extension to OLSR. It is obvious that reactive MANET routing protocols have gained much more attentions by MANET researchers. While reactive routing is more scalable and generally more beneficial when node mobility is high, its advantage over proactive approaches becomes insignificant or even negative as traffic load becomes heavier and data communication among the nodes become more widespread and more volatile, in which case the control communications for route maintenance/discovery increase exponentially in a reactive routing scheme.

The routing metric we are considering is the per-node local queuing delay, which is the dominant component of end-to-end delay. Queuing delay is very dynamic, it is determined by traffic load and available bandwidth, so it is a sign of bandwidth utilization and congestion level. Hence, queuing delay is one of the most important routing metrics to be considered in order to enhance the adaptability of the routing protocol. Note that by considering delay we are not implying any enforcement of QoS constraints on routing. Our routing system is still a best-effort routing system

We also add another level of intelligence to MANET proactive routing by predicting on queuing delays. Due to the dynamics in a MANET, any form of measured routing metric values do not necessarily reflect the future network conditions of the moment at which these values are actually used in routing decision making. Thus our predicted queuing delays are theoretically more reliable to be used as a routing metric compared to those purely measured routing metric values. We propose to use neural network as the prediction method to handle the nonlinear and non-stationary dynamics of queuing delays. Over the last two decades many neural network applications have been developed. Examples areas are data clustering (e.g., [14]), industrial control and optimization

(e.g., [15]), as well as computer networking such as call admission control [16] and ATM multiplexing [9].

The rest of the paper is organized as follows: In Section 2 we discuss some related work. The general delay prediction scheme is presented in Section 3. The neural network models used for delay prediction are discussed in Section 4. In Section 5 we present OLSR_NN, which is an extended version of OLSR with integration of our delay prediction mechanism. Specially, our TierUp routing table calculation algorithm is presented. Section 6 is dedicated to the simulation of OLSR_NN. Section 7 provides concluding remarks. Finally some future work is presented in Section 8.

# 2. RELATED WORK

## 2.1 Related MANET Routing Schemes

OLSR (Optimized Link State Routing) is perhaps the most prominent proactive MANET routing protocol to date. The cores of OLSR are its link state style routing table computation algorithm and Multi-Point Relay (MPR) technique for efficient broadcasting. Besides routing table computation, the other important functionalities of OLSR include neighbor discovery, topology dissemination, and route recovery on link failures.

Evaluating routing metrics other than hop count have been excised in quite a few extensions to existing MANET routing protocols. QOLSR [2] is one major extension in the proactive routing category. As what the 'Q' stands for, QOLSR aims at adding QoS provision to OLSR. QOLSR enables evaluation of multiple routing metrics including basically available bandwidth and delay, with higher priority on bandwidth. It has both best-effort routing algorithm and QoS routing algorithm. For best-effort routing, it searches for so called a shortest-widest path, which is the path with maximum bottle neck bandwidth and chooses the path with shortest end-to-end delay if multiple paths tie on bandwidth evaluation. For QoS routing, a heuristic method based on Lagrangian relaxation is used to find the route satisfying the QoS restrictions. The measured available bandwidth and delay of each link are used in the routing decision making. In [17], a QoS approach similar to QOLSR is presented. This approach differs from QOLSR mostly in its available bandwidth measurement and MPR selection. In contrast, our system focuses on best-effort routing rather than QoS routing.

Similar to the available bandwidth, a link throughput based metric called expected transmission count (ETX) is introduced in [6]. Each node independently measures the ETX of the link to each neighbor. The routing decisions are made such that the ETX of the route, which is the sum of the ETX of each link along this route, is minimized. The ETX of link is the inverse of the product of the delivery ratio of sending packets using that link and the reception ratio of the corresponding acknowledgements. Nodes have to use periodical link probe packets to measure the delivery ratios required for ETX calculation. Although ETX claims its design is independent of network load, but the delivery ratios it uses in calculation are actually affected by network load. Also, the added control communication overhead of ETX is significant. More over, ETX has the limitation of overestimating link delivery ratio when the data packet sizes are much larger than the link probe packet size. In a later paper [3], the same research group derived another routing metric, estimated transmission time (ETT), from ETX. In addition to link delivery ratio, ETT also handles varying nominal bandwidths of the links. ETT is designed to be use in a hybrid MANET routing scheme combining both link stake routing table computation and DSR-style on-demand querying. In contrast, our system does not conduct link metric measurements as complex as ETX or ETT. Our metric measurement is simple local operations of each node without any dedicated control communication..

## 2.2 Link State Prediction

Exponential smoothing is widely used in network protocols for estimations of specific metrics such as round trip delay in TCP. A more interesting method, so-called encountered delay estimation, is presented in [8]. The basic idea of this estimation method is to estimate the per-packet delay on a link at any given future time t as a conditional exponential-decaying function with respect to a given initial delay value measured at time $t_0$, and the estimated value changes towards a steady-state mean delay as the distance between t and $t_0$ gets longer. Each node constantly measures its local per-packet delay and estimates future per-packet delays. The estimated delays are broadcasted to the neighbors only if they are significantly different from the steady-state mean delay. A node updates its routing table upon sending out every packet, rather than upon receiving the link state updates. The way to update the routes is to sequentially estimate the delay the packet will experience (therefore so-called encountered delay) on each link along every possible route and choose the shortest one.

However, this mechanism is questionable in the following aspects. Firstly, the estimation is based on the assumption that the change of the delay series is a stationary stochastic process as we will discuss later, the queuing delay time series is actually nonstationary. Secondly, there is a lack of sound method to estimate or calculate the steady-state mean. The steady-state mean delay of each link is simply assumed to be known by every node. Thirdly, when a packet is to be sent out, every node has to estimate the encountered delay of that packet on all the links via all possible paths. The computational overhead could be very high when the traffic is heavy and the topology is complex. Lastly, the estimated link delays are path-dependent, which prevent this mechanism from smooth integration with the standard shortest-path algorithm (Dijkstra's algorithm) as what the paper claimed.

# 3. MEAN QUEUING DELAY PREDICTION

## 3.1 Prediction Target

As we mentioned in the section of introduction, we choose queuing delay as our routing metric. First of all, we have some important assumptions regarding the features of the MANET environment we dela with. We assume all the nodes share a common omnidirectional radio channel with a fixed nominal bandwidth. We also assume each node uses a simple FIFO queue for all the outbound packets, with the exemption of letting control packets having priority over data packets. These assumptions simplify the queuing delay handling, which means uniform treatment to all queued data packets disregarding their flow identities and no different bandwidths depending on which pair of local/adjacent network interfaces are used for the packet transmission. It will be interesting however more complicated to deal with queuing delay subject to varying bandwidths and

advanced queuing disciplines (e.g., Weighted Fair Queue). Research in that direction can be one of our future work.

We define the queuing delay of an outbound packet as the time span from the instant at which that packet is put on the head of the queue to the instant at which the node successfully acquire the channel for transmitting that packet. The variation of queuing delays is complicated. It is affected by the traffic load and the available bandwidth, both of which are dynamic. It is also affected by the MAC layer media access control mechanism. Researchers have applied complex statistical model [22] and Markov chain [7] to analyze the queuing behavior in a wireless network using 802.11 MAC protocol. For efficiency purpose, we measure and predict on mean (rather than individual) local queuing delays, each of which is averaged over all the outbound packets sent in a fixed-length interval.

## 3.2 Prediction Methods Comparison

Traditional statistical time series forecasting methods, including moving average, exponential smoothing, and Auto-Regressive Moving Average (ARMA), all assume stationarity of the time series. ARMA is just an extension of moving average with adjustable coefficients and error terms to allow more flexible linear generalization. In contrast, neural networks do not have any stationarity constraint on the time series to be learned. It has highly flexible nonlinear regressive structure to fit the target pattern space.

Practically, there is no guarantee that neural network can always achieve better prediction accuracy on any time series, compared to those aforementioned statistical forecasting methods. However, besides having no stationarity constraint, neural network has several other advantages over those statistical forecasting methods. To show these advantages, we conducted a comparative experiment of using neural network, moving average, and exponential smoothing to predict delays in the real Internet mean delay time series obtained from Auckland-VI delay trace file. Mathematica Neural Networks package [23] is the software providing the needed neural network functionalities. The prediction performances are shown in Figure 1.

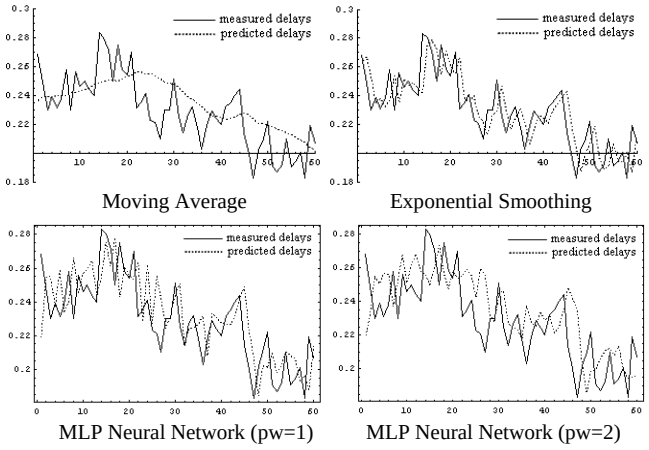



**Figure 1. Predictions on the Real Internet Queuing Delays Given by Different Prediction Methods**

As we can see, the Multi-Layer Perceptron (MLP) neural network is able to make prediction with different prediction window (PW, the number of future intervals covered by the prediction). With both prediction windows, MLP neural network made predictions better than moving average but worse than exponential smoothing. However, in practice, it is importance to notice that when incorporating the delay prediction mechanism with proactive routing, the prediction window should be always greater than one. This is because of the timing of the prediction and Topology Control (TC) message generation: the historical measured mean delays needed for prediction are not fully available until the beginning of the first interval being covered by the prediction, and the predicted delays have to wait for the next TC message dissemination to be informed to other nodes. Thus, significant part of the first interval covered by the prediction probably has passed by the time another node receives the predicted delays, and the first predicted delay becomes trivial.

Conducting prediction with different prediction windows is fairly easy in a neural network model. In contrast, moving average or exponential smoothing does not handle prediction window greater than one. Exponential smoothing always needs the most recent actual value of the time series to predict one time step ahead. Moving average could be modified to handle prediction window greater than one by substituting the needed actual value(s) with the predicted counterpart(s), but the resulting prediction smoothes out the fluctuation of the time series even more and thus fail to catch the pattern of changes of the time series.

Hence, neural network method is superior than the moving average and exponential smoothing in predicting nonstationary mean queuing delay time series

## 3.3 Mean Queuing Delay Measurement

Assuming 802.11 is the MAC protocol being used, queuing delay can be viewed as consisting of three parts: the time needed to discharge packets from the queue to a perfectly empty wireless channel, which is determined by the traffic load and the nominal bandwidth (maximum bandwidth capacity), the accumulated collision avoidance time including the time waiting for a busy channel to become clear and the backoff time, and other accumulated media access control overhead such as the time needed for channel acquirement (RTS/CTS exchange) and Distributed Inter-Frame Space (DIFS). The first part is only affected by local traffic and the fixed nominal bandwidth. The second part and the third part of queuing delay are more dynamic than the first part, they are affected by the traffic loads and the topology changes of a neighborhood involving those nodes which interference ranges cover the node of interest.

We conduct predictions on queuing delays. Besides the benefit of providing more reliable congestion information to routing decision making process happening in the near future, prediction on delays also has the potential benefit of reducing possible routing oscillation because it catches the trend of queuing delay changes rather than only the most recent queuing delay. However, prediction on the total queuing delay is very difficult due to those aforementioned dynamics. Our current solution is to rule out the influence of the random wireless channel contention and only concentrate on the first part of queuing delay, which can be easily calculated.

Our mean queuing delay calculation is based on a virtual perfect FIFO queue which dequeue rate only depends on the nominal

bandwidth regardless of the channel contention. The queue size right before a packet is enqueued is:

$$V_i = S_{i-1} - (t_i - t_{i-1}) \times B \qquad (1)$$

in which $S_{i-1} = V_{i-1} + h_{i-1}$ , and initially $V_1 = Q_1$

In Equation 1, $V_i$ is the size of the virtual FIFO queue at the time right before the $i$th packet is enqueued, $S_{i-1}$ is the size of the virtual FIFO queue at the time right after the $(i-1)$th packet is enqueued, $h_{i-1}$ is the packet size of the $(i-1)$th packet being enqueued, $t_i$ is the time at which the $i$th packet is enqueued, B is the fixed nominal bandwidth, and $Q_1$ is the actual queue size right before the first packet was enqueued during the measurement period. The mean queuing delay of the $k$th interval is then simply

$$\frac{\sum_{i \in P} V_i / B}{N_k} \qquad (2)$$

where $P$ is the aggregation of all the packets enqueued in the $k$th interval, $V_i$ is the calculated queue size using Equation (1), $B$ is the fixed nominal bandwidth, and $N_k$ is the total number of packets being enqueued in the $k$th interval.

After all, our purpose for delay prediction is to provide the routing decision maker with sound node costs information rather than precise actual queuing delays. Mean queuing delays calculated the way as shown above reflects the local traffic load and helps with balancing traffic. Thus it can be considered as an efficient routing metric, and it makes the neural network learning easier due to reduced dynamics of the time series.

## 3.4 General Delay Prediction Procedure

Briefly, the delay prediction is an adaptive neural network training and prediction process conducted by each node independently and continuously. Each node periodically measures its mean local queuing delay for each fix-length interval and uses them to train the neural network. Frequently, the neural network is retrained in order to be adapted to the network dynamics. Predicted delays are generated every interval using the trained neural network. Whenever the proactive routing protocol has a Topology Control (TC) message to broadcast, the available predicted delays are packed in the topology control message if they cover at least one interval in the future. A node receiving such a TC message retrieves the carried predicted delays and stores them as the node costs of the corresponding node to be used in routing table computation.

# 4. NEURAL NETWORK MODELS USED FOR DELAY PREDICTION

Numerous neural network models have been developed for time series prediction (e.g., [5][13]). Among them, the three major types are: Multi-Layer Perceptron (MLP) networks, Radial Basis Function (RBF) networks, and recurrent networks. MLP networks and RBF networks both belong to the category of feedforward networks. MLP networks and RBF networks conduct off-line iterative learning over a set of stored training data. Because the learning process has full control of the training data set, many methods (e.g., batch learning, multiple random initialization, etc.) can be used to optimize and speedup the learning towards the minimum of the objective function, which is some form of error evaluation. In contrast, Recurrent networks conduct on-line incremental learning, which means there is no full set of stored training data, no iterative learning is allowed, the inputs are constantly updated with the feedback from the output, and the learning proceeds indefinitely. As a result all those optimization methods usable to off-line learning can not be used in on-line learning. Compared to the training of MLP networks or RBF networks, the training of a recurrent network is generally more difficult and less reliable. This comparison does not mean that a recurrent network is generally worse than a MLP network or a RBF network, in opposite, given sufficiently large size (much larger than what is needed for a MLP network or a RBF network) of initial training data for warm-up, a recurrent neural network may generate more accurate and more adaptive prediction on a nonlinear and nonstationary time series, such as the dynamic delay time series of our interest.

For the sake of achieving efficient neural network training, we focus on investigating using MLP networks and RBF networks in our delay prediction system. The four specific neural network models are listed in Table 1.

**Table 1. Four artificial neural network models used for delay prediction**

| | Type of Inputs | Type of Feedforward Network |
|---|---|---|
| Model 1 | AutoRegressive (AR) | Multi-Layer Perceptron (MLP) |
| Model 2 | AutoRegressive (AR) | Radial Basis Function (RBF) |
| Model 3 | AutoRegressive with eXtra input signal (ARX) | Multi-Layer Perceptron (MLP) |
| Model 4 | AutoRegressive with eXtra input signal (ARX) | Radial Basis Function (RBF) |

The difference between AR and ARX regarding our delay prediction is: AR only uses lagged mean delay values as the inputs, while ARX uses both lagged mean delay values and corresponding direct affecting factor such as traffic load as the inputs. In most of the existing time series prediction scheme using neural networks, only the time series itself is taken as the input. A time series is a series of observation of the system states of a dynamical system. Taking a certain length of such a time series as the lagged input of a neural network, as pointed out in [12], are actually unfolding the state space of the underlying dynamical system. In most cases, this input is sufficient because the neural network is able to learn the pattern of a time series by appropriately unfolding the time series in temporal dimension. However, in theory, if a dominant affecting factor of the time series can be identified and measured, adding that affecting factor as an extra input can help with catching the dynamics of the time series. The problem is the effect of such a dominant affecting factor is normally short-term and does not support prediction window greater than one. Therefore we do not consider using any extra input, hence any ARX model, in our delay prediction.

The essential difference between a MLP network and an RBF network is their different approaches of approximating the target pattern surface. To decompose the target pattern surface, a MLP model uses hyperplanes, while an RBF model uses hyperspheres. This means a MLP network feeds weighted sum of the inputs to a neuron and uses sigmoid activation functions; While an RBF

network feeds a neuron with the distance between the input vector and a center vector represented by the weights and uses Gaussian activation functions. The general form of a sigmoid function is $\varphi(v_j) = 1/(1+exp(-v_j))$, where $v_j$ is the input level (the weighted sum of all the inputs) of neuron $j$. The general form of a Gaussian function is $\varphi(x) = exp(-\beta||x - c_j||^2)$, where $||x - c_j||$ is the Euclidian distance between the input vector $x$ and the basis function center $c_j$ at neuron $j$, and $\beta$ is a weight adjustable in the learning process. The radial basis activation function makes an RBF network only need one hidden layer. The order of a Gaussian function is greater than a Sigmoid function, this means in each iteration of training, a RBF network can change more aggressively to approximate the target pattern space compared to a MLP network. The upside of this aggressiveness is that given correct track, a RBF network is able to converge faster than a MLP network. The downside of it is that RBF is more prone to overfit, which means fit the training data set well but fit the data set to be predicted significantly worse.

The structural parameters of the four models are:

- Size of lagged inputs (*na*)
- Number of hidden layers (*nl*)
- Number of hidden neurons (*nh*)
- Size of the prediction window (*PW*)
- Length of a unit interval for mean delay measurement (*T*)
- Size of the measured mean delays for training (*Size_t*)

The most important parameters are the length of lagged inputs (*na*) and the number of hidden neurons (*nh*), which determine the learning capability of the neural network. It is normally sufficient to set *nh* to be slightly larger than *na*. A value close to the embedded dimension of the underlying dynamical system is a proper size of the lagged input *na*. A few methods of determining the embedding dimension ([1][12][20]) were presented by researchers. Among them, the False Nearest Neighbor method is a simple-computing trial and evaluation method.

# 5. OLSR_NN – Extending OLSR with Delay Prediction and Evaluation

## 5.1 TierUp – A Manet Node-State Routing Table Calculation Algorithm

We developed *TierUp* algorithm as a derivative of Dijkstra's algorithm to utilize the predicted delays as a routing metric in a proactive MANET routing protocol. The essential difference between *TierUp* algorithm and Dijkstra's algorithm lies on the treatment of individual costs: while Dijkstra's algorithm evaluates individual costs on link basis, *TierUp* algorithm evaluates individual costs on node basis. Although due to different levels of wireless interferences experienced by the nodes, the queuing delay of an individual packet can be link specific, in other words, affected by different channel acquirement time needed for sending packets to different neighbor nodes, our routing metric is the per-interval mean queuing delay, which statistically has no correlation with any specific links. In this sense, *TierUp* algorithm can be categorized as a node state (rather than a link state) routing table calculation algorithm.

The other important difference between *TierUp* algorithm and Dijkstra's algorithm is the reason for the name of *TierUp*. In Dijkstra's algorithm, the cost from the source node *s* to a node *v* may be updated by the smaller cost through one of *v*'s neighbors, which may have shorter, same, or even longer hop distance to *s* compared to *v*'s hop distance to *s*. Contrastively, in *TierUp* algorithm, the cost from *s* to *v* will never be updated by any smaller cost through one of *v*'s neighbors which hop distance to *s* is longer than the hop distance from *v* to *s*. If we call the nodes having the same hop distance to the source node *s* as peer nodes in the same tier, the cost from s to any node in a certain tier is not used to update the cost from *s* to any node in any lower tier. There are three purposes of this special treatment: to avoid spreading channel contention to bigger area due to taking route with longer hop distance, to reduce the probability of broken route due to going through more nodes, and to save end-to-end energy consumption.

The computational complexity of *TierUp* algorithm is less than Dijkstra's algorithm. There are two major reasons for this: Firstly, the number of additions to calculate the possible shortest distances (denoted as $N_{add}$ hereafter) in *TierUp* algorithm (i.e., the instruction d_nexthop[v] := d[v] + $w_v$) is much less than that number in Dijkstra's algorithm. For a network with *N* nodes and *L* links, $N_{add}$ for Dijkstra's algorithm is 2*L*, while $N_{add}$ for *TierUp* algorithm is *N*-1. In the case when *L* is much larger than *N*, the saving in computation is substantial; Secondly, in Dijkstra's algorithm, for each node *x* which shortest distance has been found, all its neighbors are involved in the comparison of the recorded distance and the new distance through node *x* to find possibly shorter distance, while in *TierUp* algorithm, only those neighbors in the same or higher tier of node *x* are involved in such distance comparison. However, compared to those MANET proactive routing table calculation algorithms only considering hop counts, *TierUp* algorithm is computationally more complex due to more algebraic calculations and more data structure accesses.

## 5.2 OLSR_NN Structure

Our delay prediction/evaluation system is integrated with OLSR. We call the extended version of OLSR as OLSR_NN, in which NN stands for neural network. While TierUp algorithm handles the most important issue of routing table calculation, other assembling, piping, and storage work are also needed to fulfill the integration of delay prediction and a proactive MANET routing protocol. Further, it is necessary to manage the adaptive delay prediction as a functional module within the routing protocol. Besides these functionalities, OLSR_NN remains the same as OLSR in other functional modules, including the link sensing using HELLO message exchange, TC message dissemination using MPR, and route maintenance. Our implementation of OLSR_NN is based on UM-OLSR-0.8.8 [21], which is a newly developed OLSR implementation compiling with RFC 3626.

The relationships among the functional modules of OLSR_NN are demonstrated in Figure 2. The Mean Delay Measurement module is embedded in the priqueue routine of ns-2 as a checkpoint of all the outbound packets right before they are inserted into the interface queue. To use Mathematica as a background routine from an external program such as OLSR_NN, we need to implement a coordinator using MathLink [24], which is a standard communication interface provided as an accessory tool of Mathematica. The MathLink Connector module is such a coordinator implemented as a separate static library. All the other functional modules are embedded in the OLSR routines.

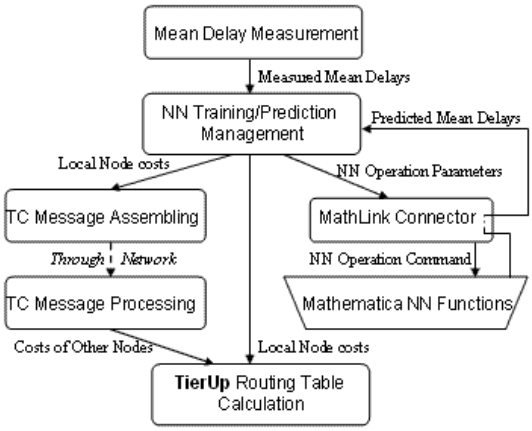



**Figure 2. Relationships among the Functional Modules of OLSR_NN**

The TC message format is extended to include the predicted delays as node costs. Two more fields, *start_time* and *node_costs*, are added. *start_time* is the starting time of the period covered by the carried predicted delays. The covered period ends at *start_time* + *PW* × *T*. *start_time* has the resolution of 0.1 second. It occupies the 16 bits reserved for future use in the original TC message format, thus it takes no extra space. *node_costs* is the vector of node costs of size *PW*. The node cost can record maximum delay of 25.5 ms as a 8-bit integer, which is more than enough in most cases. By doing so, the extra TC messaging overhead is only *PW*×8 bits, and the routing agent can use node costs evaluated at 256 levels from 0 ms to 25.5 ms.

# 6. SIMULATION

## 6.1 Experimental Design

We conducted extensive simulations on OLSR_NN. We used ns2-2.29 as the simulation tool. Since it is more valuable to test the adaptability of OLSR_NN to heavy traffic rather than light traffic, we used 14 packet flows among randomly selected nodes, the average packet rate of each flow is set to 40 kbps or 50 kbps for different test scenario. Among the 14 packet flows, 12 of them are in exponential pattern and the other 2 are in CBR pattern. Exponential and CBR are two popular traffic patterns and they fit the statistical characteristics of most networking applications. The combination of them produces a fairly complex overall traffic pattern that challenges the adaptability of the routing protocol being tested.

We tried four levels of node mobility with node moving speed of 2 m/s, 5 m/s, 10 m/s, and 15 m/s respectively, with 5 or 10 seconds of pausing time is imposed randomly between two consecutive node movements. The node movement pattern is Random Waypoint model. We used Random Trip Mobility Models [4] to generate different node mobility scenarios. This model has the merit of initializing scenario with a steady state rather than a transient state which causes unstable routing performance.

We set the network area as 1650×1400 $m^2$. We observed in the simulation that a node can detect remote nodes of normally up to 6 to 9 hops away, with the maximum of 12 hops away in some time. With topology complexity at this level, routing is certainly not a trivial problem. The other important fixed parameters include: 50 nodes, transmission range of 250 m, nominal channel bandwidth of 2M bps, packet size of 512 bytes, outbound packet queue capacity of 50 packets, and simulation time of 800 seconds.

The fixed parameters specific for OLSR_NN include: HELLO message interval of 2 seconds, TC message interval of 5 seconds, which is also set as the delay measurement interval. For the sake of efficient training, MLP neural network model is used in all the experiments.

The prediction window size was set to 3 in most of the time and 2 otherwise. Since the predicted values are disseminated to other nodes through TC message broadcast, which is lossy, smaller prediction window probably causes significant lack of up-to-date predicted metric values for routing table computation. On the other hand, the farther the prediction reaches, the less accurate the prediction is, and larger prediction window incurs larger TC message size. We found that three is an appropriate prediction window size to balance such a tradeoff in most of the time.

## 6.2 Simulation Results

In almost all the eight scenarios (see Figure 3), OLSR_NN is able to reduce average end-to-end delay and increase data packet delivery ratio compared to OLSR. Each error bar in these two graphs represents the standard deviation of ten runs of a specific testing scenario, with non-overlapping streams of random numbers generated for each run. Since data packet delivery ratio is the most important routing performance criteria, in this section we only concern those test scenarios rendering the best data packet delivery ratio among all the test scenarios we conducted. Since more successful packet delivery generally increases the chances of prolonged delay, there is no guarantee that the decreases of average end-to-end delay shown in Figure 3 are also the most among all test scenarios. Figure 3 also demonstrate the benefit of OLSR_NN using MLP neural network for prediction (OLSR_NN) over OLSR with delay prediction using two simple statistical prediction methods: moving average (OLSR_MA) and exponential smoothing (OLSR_EXP). OLSR_NN achieved the highest delivery ratio and lowest average end-to-end delay in almost all the test scenarios. OLSR_MA and OLSR_EXP in general render better performances compared to OLSR, but worse than OLSR_NN.

The overall improvements in data packet delivery ratio and average end-to-end delay indicate that OLSR_NN is able to balance the traffic significantly better than OLSR, therefore reduce the congestion level. It can be observed that the routing performance improvements in scenarios having heavier traffic are more than those in scenarios having lighter traffic. The main reason, as our analysis revealed, is that the room for traffic balancing in lighter traffic scenarios is less than that of heavy traffic scenarios. This smaller room for improvement has two aspects. Firstly, due to significantly reduced traffic, there is normally more room in the interface queues, which enhances the capability of accommodating bursty traffic due to biased routing without dropping packets. Secondly, the traffic distribution is evener in lighter traffic scenarios, which helps with alleviating the congestions. Another interesting observation is that less performance improvements were achieved as node mobility became higher. We infer the reason is that higher node mobility

increases topology instability, which causes more link failures and there are less alternative more stable links to choose.

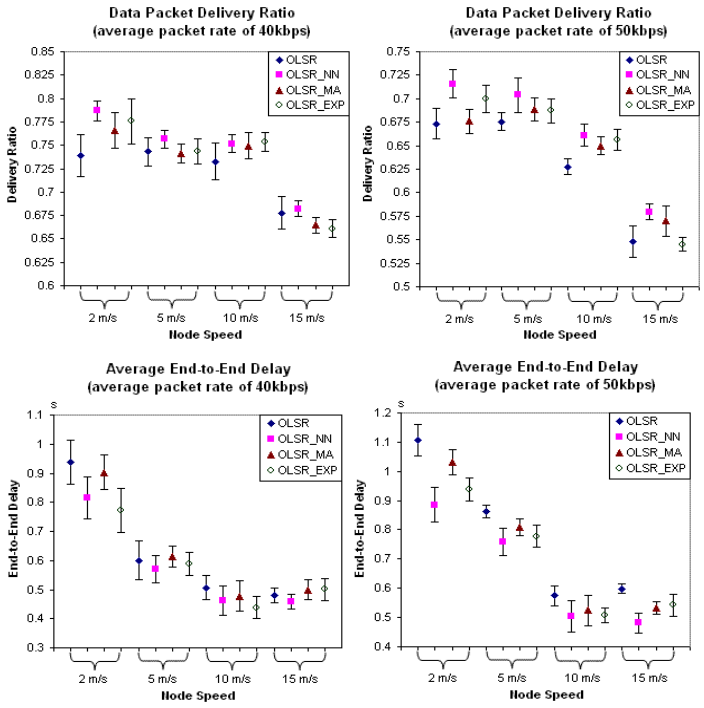
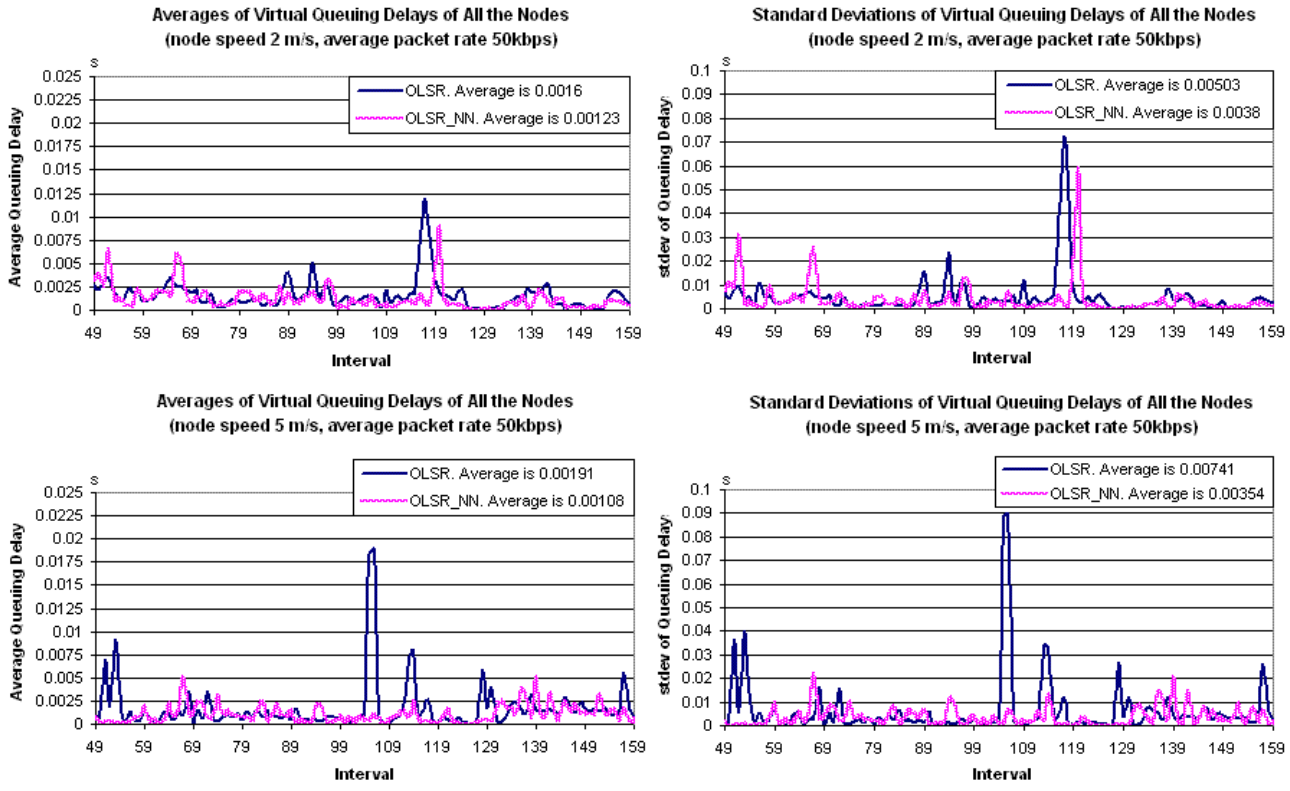



**Figure 3. Routing Performance Comparison between OLSR, OLSR_NN, and OLSR with delay prediction using Moving Average (OLSR_MA) and Exponential Smoothing (OLSR_EXP)**

The mean queuing delay of a node in a particular interval reflects the traffic load of that node in that interval. Thus the mean queuing delays of all the nodes in a particular interval reflect the traffic distribution in that interval. The relative level of traffic balance can then be evaluated by comparing the averages and standard deviations of per-interval mean queuing delays of all the nodes. Figure 4 illustrates such statistical analysis on OLSR and OLSR_NN in the heavier traffic scenarios, in which OLSR_NN achieved better routing performance improvements. Due to scaling difficulty, a few large abrupt values are not shown in Figure 4. To count in those values, the averages of the statistical values are calculated and shown in the legends of the plots in Figure 4.

Most of the simulation time is covered in the statistical analysis for each scenario. We chose to start the statistics from the 49th interval, or 5×(49-1) = 240th second, which is the starting time at which *TierUp* algorithm take effect. Recall that we set the neural network initialization time to 225th second, and we allow three intervals for the nodes to collected needed predicted delays before starting executing *TierUp* algorithm for routing table computation. OLSR_NN has no effect to the routing before the 49th interval.

As shown in Figure 4, OLSR_NN achieves shorter average delays compared to OLSR in all four scenarios. More importantly, significantly smaller average standard deviations are achieved, which indicates more evenly distributed traffic and contributes the most to the routing performance improvements.



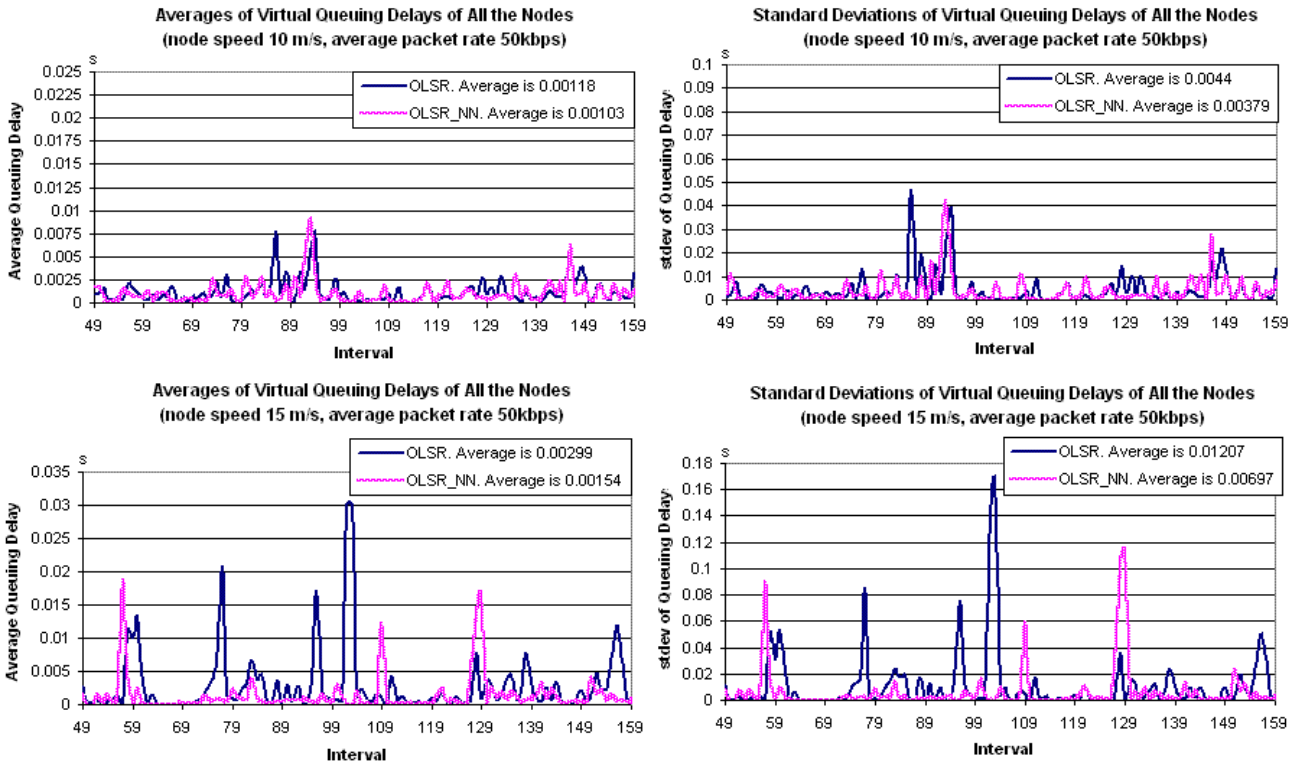



**Figure 4: the Change of Averages and Standard Deviations of Per-interval Measured Mean Queuing Delays of All the Nodes. Each set of plots compare such statistics between OLSR and OLSR_NN in heavier traffic (50kbps packet generation rate in average) with the same particular node mobility.**

It is important to notice that the traffic balance is evaluated for each interval rather than consecutive multiple intervals. The traffic load changes over time are determined by the packet generation patterns of the nodes generating the traffic. The routing agents can only try to balance the accumulated traffic (newly generated plus leftover) in every short interval. They do not have the capacity and the knowledge to balance the traffic over time. Hence, in Figure 4, only the magnitude of the values matters, not the fluctuation level.

## 7. CONCLUSION

The key of the integration of our delay prediction mechanism and a proactive MANET routing protocol is to enable utilization of the predicted delays in routing table calculation. To achieve this, we developed a node state based algorithm called *TierUp*, which is a light-weighted derivative of Dijkstra's algorithm. We integrated our delay prediction mechanism and *TierUp* algorithm into OLSR and call this extended version of OLSR as OLSR_NN.

Our simulation shows that for a network with big range of node mobility, OLSR_NN can increase data packet delivery ratio and reduce average end-to-end delay significantly, at the mean time maintain the overall throughput at the same level or even improve it. The routing performance improvement brought by OLSR_NN is immune to node mobility change as long as there is substantial room for traffic balancing. Hop count dependent MANET routing protocol, such as OLSR, generally causes more biased routing in heavier traffic scenarios compared to in lighter traffic scenarios. Lighter traffic also leaves more room in the queues of the nodes to accommodate bursty traffic. Therefore there is more room for OLSR_NN to balance traffic in heavier traffic scenarios compared to lighter traffic scenarios. Finally, the flexibility of key parameter configuration makes OLSR_NN adaptive to diverse networking conditions.

## 8. FUTURE WORK

First of all, there is still substantial room for prediction accuracy improvement by using more advanced neural network technologies, such as recurrent neural networks, as they are substantiated. With more powerful neural network prediction capability, the actually queuing delays, which include the part of delay caused by channel contention, may be predicted with sufficient accuracy, therefore provide a more reliable view of the congestion conditions to the routing decision maker. Another direction would be using some heuristics based on locally available topology or traffic information to estimate the relative level of the part of queuing delay caused by channel contention. Nevertheless, the added computational overhead of these approaches should be carefully controlled.

Another valuable future work is to conduct emulation on our MANET proactive routing system integrated with delay prediction mechanism. Emulation on real network nodes in the distributed manner will radically release the computational burden of a workstation running network simulation. We are also hoping that emulation can render more realistic performance evaluation compared to ns-2. Event scheduling in ns-2 is problematic. It has no control to the order of the executions of networking activities

relative to the executions of those relevant algorithmic activities. Therefore it is very likely that some networking activities are fired prematurely before those relevant algorithmic activities, which are supposed to be done first, are completed.

## REFERENCES


[1] Abarbanel, H. D. I., Brown, R., Sidorowich, J. J. and Tsimring, L. Sh. *The analysis of observed chaotic data in physical systems*, Reviews of Modern Physics, vol. 65, no. 4, pp. 1331-1392, October 1993

[2] Badis, H. and Agha, K. A., *QOLSR, QoS routing for ad hoc wireless networks using OLSR*, European Transactions on Telecommunications, vol. 15, no. 4, 2005.

[3] Bicket, J., Aguayo, D., Biswas, S., and Morris, R., Architecture and Evaluation of an Unplanned 802.11b Mesh Network, *in proceedings of the 11th Annual International Conference on Mobile Computing and Networking 2005 (MobiCom'05)*, Cologne, Germany, August 28 - September 2, 2005

[4] Boudec, J.-Y. L. and Vojnovic, M. Perfect simulation and stationarity of a class of mobility models, *in Proceedings of IEEE Infocom 2005*, Miami, FL, 2005

[5] Cai, X., Zhang, N., Venayagamoorthy, G. K., and Wunsch II, D. C., Time series prediction with recurrent neural networks using a hybrid PSO-EA algorithm, *in proceedings of International Joint Conference on Neural Networks (IJCNN'04)*, Budapest, Hungary, July 2004

[6] De Couto, D. S. J., Aguayo, D., Bicket, J., and Morris, R. A high-throughput path metric for multi-hop Wireless routing, *in proceedings of the 9th Annual International Conference on Mobile Computing and Networking 2003 (MobiCom'03)*, San Diego, California, September 14-19, 2003

[7] Engelstad, P. E. and Østerbø, O. N., Queuing delay analysis of IEEE 802.11e EDCA, *in proceedings of 3rd Annual Conference on Wireless On-demand Network Systems and Services*, Les Ménuires, France, January 18-20, 2006

[8] Eom, H., *Improving link-state routing - by using estimated future link delays (revised)*, Technique Report CS-TR-4297R, Computer Science Department, University of Maryland, College Park, MD, December 2002

[9] Habib, I. W., Tarraf, A. A., and Saadawi, T. N., *A Neural Network Controller for Congestion Control in ATM Multiplexers*, Computer Networks and ISDN Systems, vol. 29, no. 3, pp. 325-334, February 1997

[10] Jacquet, P., Muhlethaler, P., Clausen, T., Laouiti, A., Qayyum, A., and Viennot, L. Optimized link state routing protocol for ad hoc networks, *in Proceedings of IEEE International Multi Topic Conference, Technology for the 21st Century*. 2001

[11] Johnson, D. B., Maltz, D. A., and Hu, Y. C., Internet-draft: The Dynamic Source Routing (DSR) Protocol for mobile ad hoc networks, *Network Working Group, Internet-draft,* July 2004.

[12] Kennel, M. B., Brown, R., and Abarbanel, H. D. I. *Determining embedding dimension for phase-space reconstruction using a geometrical construction*, Physical Review, vol. 45, no. 6, March 1992

[13] Lendasse, A., Oja, E., Simula, O., and Verleysen, M., Time series prediction competition: the CATS benchmark, *in proceedings of International Joint Conference on Neural Networks (IJCNN'2004),* Budapest, Hungary, July 2004

[14] Malakooti, B. and Raman, V., *Clustering and selection of multiple criteria alternatives using unsupervised and supervised neural networks*, Journal of Intelligent Manufacturing, vol. 11, pp. 435-453, 2000

[15] Malakooti, B. and Raman, V., *An interactive artificial neural network approach for machine set-up optimization*, Journal of Intelligent Manufacturing, vol. 11, no. 1, pp. 41-51, 2000

[16] Morris, R. J. T., and Samadi, B., *Neural Network Control of Communications Systems*, IEEE Transactions on Neural Networks, vol. 5, no. 4, pp. 639-650, July 1994

[17] Nguyen, D. and Minet, P., QoS support and OLSR routing in a mobile ad hoc network, *in proceedings of the International Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies (ICNICONSMCL'06)*, April 23 - 29, 2006

[18] Ogier, R., Templin, F., and Lewis, M., RFC 3684 - Topology Dissemination Based on Reverse-Path Forwarding (TBRPF), *Network Working Group, Request for Comments: 3684,* February 2004.

[19] Perkins, C., Belding-Royer, E., and Das, S., RFC 3561 - Ad hoc On-Demand Distance Vector (AODV) Routing, *Network Working Group, Request for Comments: 3561,* July 2003

[20] Pi, H., and Peterson, C., *Finding the embedding dimension and variable dependencies in time series*, Neural Computation, vol. 6, no. 3, pp. 509-520, May 1994

[21] Ros, F. J., UM-OLSR, an implementation of the OLSR (Optimized Link State Routing) protocol for the ns-2 network simulator. Available: http://masimum.dif.um.es/?Software:UM-OLSR

[22] Tickoo, O. and Sikdar, B., Queuing analysis and delay mitigation in IEEE 802.11 random access MAC based Wireless networks, *in proceedings of INFOCOM 2004. Twenty-third AnnualJoint Conference of the IEEE Computer and Communications Societies,* vol. 2, March 7-11, 2004 Page 1404 – 1413

[23] Wolfram Research, Mathematica Neural Networks. Available http://www.wolfram.com/products/applications/neuralnetworks/

[24] Wolfram Research, MathLink. Available: http://www.wolfram.com/solutions/mathlink/mathlink.htm