A self-adaptive routing paradigm for wireless mesh networks based on reinforcement learning

Conference Paper · January 2011
DOI: 10.1145/2068897.2068932 · Source: DBLP

Available from: Luciano Lenzini
Retrieved on: 10 May 2016
A Self-Adaptive Routing Paradigm for Wireless Mesh Networks Based on Reinforcement Learning

Maddalena Nurchis  
IMT Lucca, Italy  
m.nurchis@imtlucca.it

Raffaele Bruno,  
Marco Conti  
IIT - CNR, Pisa, Italy  
{r.bruno,m.conti}@iit.cnr.it

Luciano Lenzini  
Dept. of Information Eng.,  
University of Pisa, Italy  
l.lenzini@iet.unipi.it

ABSTRACT

Classical routing protocols for WMNs are typically designed to achieve specific target objectives (e.g., maximum throughput), and they offer very limited flexibility. As a consequence, more intelligent and adaptive mesh networking solutions are needed to obtain high performance in diverse network conditions. To this end, we propose a reinforcement learning-based routing framework that allows each mesh device to dynamically select at run time a routing protocol from a pre-defined set of routing options, which provides the best performance. The most salient advantages of our solution are: i) it can maximize routing performance considering different optimization goals, ii) it relies on a compact representation of the network state and it does not need any model of its evolution, and iii) it efficiently applies Q-learning methods to guarantee convergence of the routing decision process. Through extensive ns-2 simulations we show the superior performance of the proposed routing approach in comparison with two alternative routing schemes.

Categories and Subject Descriptors


General Terms

Performance, Algorithms.

Keywords

Wireless mesh networks, unicast routing, opportunistic routing, reinforcement learning, performance evaluation.

1. INTRODUCTION

In recent years wireless mesh networks (WMNs) have attracted increasing attention due to their advantages over traditional wireless networking technologies. In particular, WMNs allow a fast, easy and cost-effective network deployment in a large variety of environments. Furthermore, WMNs are able to provide connectivity services even when fixed-network deployments are limited or not available at all. For instance, WMNs have been used as a communication infrastructure for wireless ISPs in rural or under-developed areas, municipal access networks, and neighbourhood communities, as well as to provide wireless communications among first responders in emergency situations, or backhaul services for large-scale wireless sensor networks [4]. Such diversity in applications and usage scenarios for WMNs generates a pressing demand for the development of mesh networking solutions characterized by a high degree of flexibility and adaptability.

Among the several issues that may affect the overall performance of a WMN, traffic routing is one of the most critical. Thus, there is a large body of studies that have proposed different routing strategies to cope with the challenges due to multi-hop operations, unreliable wireless links, unpredictable channel interference, location-dependent contention, uncertain traffic demands, QoS requirements, etc.. At one end of the spectrum there are unicast routing solutions. Many routing protocols in this class select minimum-cost path according to a given routing metric (e.g., [10] and references therein). Alternatively, path computation can be formulated as an optimization problem (e.g., [11, 15]). On the other end of the spectrum there are opportunistic routing solutions [6], which broadcast each packet and implement forwarding decisions in a hop-by-hop fashion. In this way, the routing protocol can leverage transmissions that unexpectedly reach far nodes, thus minimizing the number of hops needed to deliver a packet to the destination.

In this paper we are not concerned with the analysis of which are the scenarios where one specific routing solution for WMNs shows better performance than another specific routing solution. On the contrary, the central question addressed in this paper is: How to design self-adaptive mesh networking solutions that autonomously and on-the-fly decide upon the best routing protocol for a traffic flow from a set of supported algorithms, given an estimate of the current network state? In principle an exact answer to this question could be obtained through a comprehensive routing model characterizing the impact of routing primitives, traffic patterns, network topologies and link characteristics on the network performance. As a matter of fact, theoretical analysis of the capacity of wireless mesh networks has received much attention in recent years (e.g., [16] and references therein). However, these studies are either limited...
to asymptotic bounds or based on centralized and computationally complex models. Furthermore, existing results cannot be applied to heterogeneous environments, where each node may use a different routing strategy.

To answer the above questions, in this paper we propose a novel routing architecture that exploits reinforcement learning to allow each node to autonomously choose the best forwarding strategy from a pre-defined set of routing protocols as network conditions (e.g., traffic loads or link qualities) change. The idea of using machine learning techniques to solve specific routing problems in ad hoc networks is not new (see [9] for a survey). However, to the best of our knowledge there is very little research in applying reinforcement learning for the optimal composition of different routing strategies. The rationale behind the use of reinforcement learning is that it enables the design of decision agents that learn through trial-and-error interactions with a dynamic environment how to behave in order to maximize the reward associated to their actions [13]. For the purpose of evaluation, in this paper we design an intelligent agent able to combine a traditional unicast routing protocol with an opportunistic routing scheme. A recent paper [3] has investigated how to combine on-demand routing and delay-tolerant routing in intermittently-connected ad hoc networks. However, the solution proposed in [3] is basically an extension of the AODV routing protocol [12] to cope with network partitions. On the contrary, the approach proposed here is more general because it allows the combination of multiple and arbitrary routing algorithms without making any assumption on the characteristics of the underlying network.

In the following, we first formulate the network state space for our decision problem with the objective of identifying the minimum number of state variables required to estimate the routing efficiency. Second, we define a set of rules to instruct the learning agent how to behave in order to maximize its reward. It is important to note that the reward function is used to formalize the goal of the learning problem. For instance, the reward function can give higher value to policies that improve throughput performance, reduce delays or increase reliability. This results in a highly configurable system able to customize the routing behaviour to different application objectives and QoS requirements. Finally, the tool we utilize to solve the learning problem is the Q-learning algorithm, which is a dynamic programming method that works by continuously improving its estimates of the value of particular actions at particular states [14].

The key properties of Q-learning are: i) it is a model-free method, i.e., it does not require any model of the system, ii) it converges with probability one to the optimal policy under general assumptions, and iii) it is very easy to implement using lookup tables. On the negative side, it may converge slowly to the optimal policy depending on the state size. Thus, to improve efficiency and speed up convergence we rely on compact network-state representations and smart initialization of action-value functions.

To confirm the effectiveness of the proposed routing approach, we present the results of an extensive simulation study using ns-2. In this study we compare the performance of three routing protocols: i) OLSR [7], which is one of the most representative and popular proactive routing protocols for WMNs; ii) PacketOPP [5], which is a recently proposed probability-based opportunistic routing protocol; and iii) Hybrid, a novel routing protocol obtained by applying the proposed reinforcement learning approach to dynamically combine OLSR and PacketOPP. Our results show that routing hybridization can lead to significant throughput improvements over single routing schemes in a broad range of network and traffic conditions.

The rest of this paper is organized as follows. Section 2 overviews the principles of reinforcement learning, and introduces the Q-learning algorithm. Section 3 describes the proposed routing architecture and the design of the routing learning agent. In Section 4 we report the results of our simulations. Section 5 concludes the paper with final remarks.

2. BACKGROUND ON REINFORCEMENT LEARNING

Reinforcement Learning (RL) is a popular machine learning technique, which allows an agent to automatically determine the optimal behaviour to achieve a specific goal based on the positive or negative feedbacks it receives from the environment after taking an action [13]. More formally, let us assume that the interactions between the agent and the environment occur at a sequence of discrete time instants $t$. Following the same notation as in [13], the learning problem can be formulated by defining:

- The state $s_t \in S$ of the environment as observed by the agent.
- The action $a_t \in A(t)$ chosen by the agent.
- The probabilistic reward $r_{t+1} = R(s_t, a_t)$, which measures the goodness of taking action $a_t$ in state $s_t$.
- The state transition function $T(s_t, a_t, s_{t+1})$, which provides the probability of making a transition from state $s_t$ to state $s_{t+1}$ after performing action $a_t$.

Typically, in machine learning it is assumed that the state transition probabilities satisfy the Markov property, i.e., they are independent of any state or action previous to time $t$. In this case, the environment where the agent operates is described through a Markov Decision Process (MDP), and several optimized learning algorithms have been studied for this class of environments\(^2\). More specifically, let us denote with $\pi$ a policy, which defines the probability $\pi(s, a)$ that the learning agent takes action $a$ when in state $s$. In some cases, a policy can be a simple deterministic function, but arbitrarily sophisticated policies are possible. In general, the main goal of the learning agent is to find the optimal policy providing the maximum long-term reward. To express this in a mathematical setting we define the expected return $W_t$ as a function of the sequence of rewards received after time step $t$. In the case of learning tasks that are continuous a typical formulation for the return functions is as follows

$$W_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1},$$

where $\gamma$ is a parameter ($0 \leq \gamma < 1$) called discount rate, used to determine the present value of future rewards. If

\(^1\)Delay-tolerant routing is a different type of opportunistic routing, where transmission opportunities are generated by node mobility rather than wireless diversity.

\(^2\)It is important to note that RL can also deal with non-Markovian environments [13]. However, Markov property is a good approximation for many network characteristics observed over long time intervals.
The solution of any RL problem can be formulated mathematically in a MDP perspective and under the discounted infinite horizon optimality model described in (1) by introducing the concept of optimal value $V^*(s)$ for each state $s \in S$. More precisely, $V^*(s)$ is defined as the expected return if the agent starts at state $s$ and then executes the optimal stationary policy $\pi^*$. From the previous definition it follows that

$$V^*(s) = \max_a E \left( \sum_{t=0}^{\infty} \gamma^t r_t \right). \tag{2}$$

In the MDP case, it is a consolidated theoretical result that the optimal value function (2) is unique and it can be computed with closed-form equations unique and it can be computed with closed-form equations if $R(s, a)$ and $T(s, a, s')$ are known [13]. However, the main idea behind RL is that it is possible to obtain the optimal policy for an MDP environment even when the model of the environment is not known or difficult to learn. Thus, several model-free RL methods have been developed, which are based on the concept of action-value functions [13]. More formally, the optimal action-value function $Q^*(s, a)$ is the expected return if the agent starts at state $s$, executes action $a$ and follows the optimal policy $\pi^*$ hereafter. A fundamental development in the context of model-free RL methods is the Q-learning algorithm [14], which updates the action-value function using the following rule

$$Q(s, a) = Q(s, a) + \alpha \left[ r + \gamma \max_{a'} Q(s', a') - Q(s, a) \right], \tag{3}$$

where $s'$ and $r$ are the state and reward, respectively, obtained after performing action $a$ in state $s$, and $\alpha$ is a positive step-size parameter determining the learning rate. It has been shown that under general assumptions\(^3\) the Q-learning algorithm will asymptotically converge with probability one to the optimal action-value function $Q^*$ independently of the agent’s behaviour, i.e., of the policy $\pi$ being followed [14].

The great advantage of Q-learning algorithm is that it is typically easier to implement than other RL techniques. On the negative side, it may converge slowly to the optimal policy due to the so-called exploration and exploitation issue. Basically, when in state $s$ the learning agent should exploit its accumulated knowledge of the best policy to obtain high rewards, but it must also explore actions that it has not selected before to find out a better strategy. Moreover, each action must be tried many times to gain a reliable estimate of its expected reward. In Section 3.2 we will discuss suitable action selection strategies for the application scenario addressed in this work.

3. ROUTING PROTOCOL DESIGN

In this section we first provide a high-level overview of our routing framework, and then we describe in detail the proposed learning-based routing system.

3.1 Routing architecture

Figure 1 shows the main components in the node architecture. A key assumption in our routing framework is that each mesh node implements a number of pre-defined routing algorithms, and it can decide on-the-fly which protocol to activate. The decision on which routing strategy to use can be made at different levels of granularity. For instance, the source node may decide the best routing algorithm on a flow-level basis. A finer granularity in the optimization process would be obtained by using a combination of routing algorithms for each connection, for instance dividing the packet flow into batches and choosing the best routing algorithm on a batch-level basis.

The core component in the architecture depicted in Figure 1 is the reinforcement learning agent implementing the Q-learning algorithm. Such agent operates between the transport layer and the network layer. Specifically, the agent receives feedback signals from the network, which are used both to estimate the network state, and to compute the (positive or negative) reward generated by the last action taken by the agent. Such feedback signals can be in the form of observed network quantities, or explicit messages sent by the destination and intermediate nodes. It is important to note that the reward function formalizes the goal of the agent and it is application dependent. For instance, in case of applications generating bulk data transfers, throughput is the most important performance metric and the reward should be a function of the measured connection throughput. In case of delay-sensitive traffic alternative formulations for the reward would be needed. From the knowledge of network states and transition rewards, the agent can update the Q-value for the last transition using formula (3). Then, the Q-values are represented by a two-dimensional lookup table indexed by state-action pairs. Note that lookup tables are very easy to implement, although the memory requirements may be prohibitive in case of huge state spaces.

Finally, there is the action selector, which decides the fraction of packets in a flow that should be forwarded using one of the available routing protocols. The simplest solution would be to always select the routing protocol that has provided the highest throughput up to the time of the agent decision. However, there is a trade-off between short-term gains and long-term rewards. Furthermore, the agent has to execute each state-action pair infinitely often in order to guarantee the convergence to the optimal $Q^*$ matrix. In following section suitable exploitation strategies will be defined.

\(^3\) The most important condition is that all state-action pairs are visited infinitely often.
3.2 Network state, reward and actions

In the rest of this section, to simplify the protocol presentation, we assume that each mesh node implements two different routing strategies, namely, an unicast routing protocol and an opportunistic routing protocol.

In our model, an action taken by the source is one possible choice for the \( \eta \) parameter, defined as the fraction of packets in batch \( i \) to be sent using the unicast routing protocol (obviously, the remaining \((1-\eta)\) packets are sent using the opportunistic routing protocol). In other words, the learning agent at the source node dynamically adapts the portion of traffic sent using unicast routing or opportunistic routing depending on its view of the network state and the received network feedbacks. More specifically, let \( \rho^U_i \) be the throughput of packets from batch \( i \) delivered using the unicast routing protocol, while let \( \rho^O_i \) be the throughput of packets from batch \( i \) delivered using the opportunistic routing protocol\(^4\). Thus, \( \rho_i = \rho^U_i + \rho^O_i \) is the total throughput for packets of batch \( i \) measured at the destination. In addition, let \( \eta_i^U \) be the fraction of the overall throughput at the destination for batch \( i \) due to the unicast transmissions. More formally, this can be expressed as \( \eta_i^U = \rho^U_i / \rho_i \). Now, we can introduce the concept of routing efficiency \( \epsilon_i \), a 3-valued variable defined as follows

\[
\epsilon_i = \begin{cases} 
1 & \eta_i^U > \eta \\
0 & \eta_i^U = \eta \\
-1 & \text{otherwise}
\end{cases}
\]  

(4)

In other words, the \( \epsilon_i \) quantity measures if the packets delivered using the unicast routing protocol suffer from less, the same number, or more losses than packets of the same batch delivered using the opportunistic routing protocol. Then, the network state \( s_i \) is the two-dimensional tuple \( \{ \eta_i, \epsilon_i \} \).

Note that our objective is not to obtain global network information, but to define an approximate compact representation of the network state to infer the impact of agent’s actions on the performance of each traffic flow.

The last element of our reinforcement learning system is the reward \( R(s_i, a_i) \) gained when in state \( s_i \), the agent performs action \( a_i \). Intuitively, for bulk data transfers the immediate reward should depend on the total throughput obtained at the end of the transmitted batch of packets. More precisely, let us assume that the source receives at the end of the transmission of batch \( i \) a feedback message with the \( \rho^U_i \) and \( \rho^O_i \) measures\(^5\). Then, the agent chooses a new value for the proportion \( \eta_{i+1} \) between unicast and opportunistic transmissions in the next batch \( (i+1) \) (see Section 3.3 for a detailed discussion on the action selection policies). At the end of the \((i+1)\)th batch, the agent will receive a new feedback message and it will calculate the reward as follows:

\[
R(s_i, a_i) = \rho_{i+1} + \omega(\rho_{i+1} - \rho_i),
\]  

(5)

where \( 0 < \omega < 1 \), is a constant parameter. As expressed in formula (5), the reward consists of two contributions. The first term is intuitive: the higher the measured throughput, the higher is the reward gained by the source node. However, the agent should also assign higher value to actions that provide an increase in the flow throughput, and penalize actions that decrease the throughput. Thus, the reward should also depend on throughput gain \( (\rho_{i+1} - \rho_i) \)\(^6\).

Whenever the source node receives a feedback message from the destination or a neighbour node with the information needed to compute the reward associated to its last action, it updates its estimates of the action values using formula (3). It is important to note that for learning problems in stationary environments it is usual to set the learning rate \( \alpha \) as a decreasing function of time. The reason is that at the beginning of the agent’s life the rate \( \alpha \) should be large enough to quickly overcome initial conditions. As the estimates of the optimal action values converge, decreasing the rate \( \alpha \) mitigates unnecessary fluctuations. However, the routing problem considered in this work is intrinsically non-stationary because the network conditions can change over time. In such cases it is useful to give more weight to recent rewards than past ones to guarantee a fast adaptation to dynamic network behaviours. This is achieved by using a constant learning rate (in our implementation \( \alpha = 0.5 \), which provides a good tradeoff between reactivity and stability). For the same reason, we also set the discount rate \( \gamma \) equal to zero, to give more importance to recent rewards than to future ones, which may be associated to a network with changed conditions.

Before concluding this section, it is important to explain how our learning agent reduces the number of admissible state-action pairs to explore. First of all, we restrict ourselves the discrete \( \eta \) values. More formally, let \( (K+1) \) denote the maximum number of positive and equally distributed values that \( \eta \) can take in the range \([0, 1]\). By definition it follows that:

\[
\eta = k\delta \quad \text{with} \quad k = 0, 1, 2, \ldots, K,
\]

(6)

where \( \delta = 1/K \). Furthermore, for each \( \eta \), we can have three different conditions for the efficiency parameter \( \epsilon \), i.e., \(-1, 0, 1\). This implies that the overall size of the state space \( S_t \) is \( 3 \times (K+1) \). In our implementation we have set \( K = 10 \), which means that the fraction of packets in a batch that is transmitted using unicast routing changes with a step size equal to 10 percent. Our results indicate that larger \( K \) values have negligible impact on system performance.

As described in Section 3.1, in our model an action is the selection of a new \( \eta \) value for the successive batch of packets. In principle, after each packet batch, the learning agent could select any of the admissible values of \( \eta \) in the range \([0, 1]\). To restrict the number of possible state-action pairs we assume that in state \( \{ \eta_i, \epsilon_i \} \) the learning agent can only increase/decrease the \( \eta \) parameter by \( \delta \), or maintain it constant. In other words, the agent’s actions can only cause transitions to adjacent states. This means that the total number of state-action pairs per each flow destination that we should rate with a Q-value is bounded by \( 27 \times (K+1) \). Note that this number is small enough to make acceptable the use of lookup tables for implementing the Q-learning algorithm.

3.3 Action selection policy

A very important learning procedure is the action selection policy. In following simulations three intuitive and quite popular policies will be tested.

\(^4\)We assume that a tiny header is added to the traditional routing header containing information such as the batch id and the type of routing algorithm used to forward that packet.

\(^5\)For the sake of simplicity we assume that the feedback signal is not delayed. In Section 3.4 we discuss on the impact of delayed feedbacks, which is the normal case in real-world networks.

\(^6\)In our simulations \( \omega = 0.5 \).
• **greedy**: The greedy method is the simplest action selection rule because, at time \( t \) it always chooses the greedy action \( a^* \), namely the action with the maximum estimated action-value function. More formally this can be expressed as follows:

\[
Q(s_t, a) = \max_a Q(s_t, a) ,
\]

where \( s_t \) is the network state at time \( t \). This method is guaranteed to maximize the immediate reward but it ignores the exploration of actions with non-maximum \( Q \) value, although they could lead to action selections with better long-term return.

• **\( \varepsilon \)-greedy**: A simple alternative to the pure greedy strategy is a near-greedy selection methods. More precisely, the learning agent will behave most of the time greedily, selecting an action with maximum expected value with probability \((1-\varepsilon)\), but with a small probability \( \varepsilon \) instead selects an action uniformly over the set \( A_t \), and independently of the action-value estimates. The advantage of an \( \varepsilon \)-greedy method is that it guarantees that every action will be sampled an infinite number of times as the time goes to infinity, which is the necessary condition for the convergence of the Q-learning algorithm. However, the advantage of \( \varepsilon \)-greedy over greedy methods highly depends on the network behaviour. For instance, if the variance of reward samples is very low, exploration may not be necessary to find the optimal action.

• **softmax**: One drawback of the \( \varepsilon \)-greedy method is that it has the same probability of choosing worst-case actions as well as best-case actions when in exploration phase. An obvious solution for this issue is to use softmax action selection rules, which rank and weight actions according to their value. The most common softmax function used in reinforcement learning to convert values in action probabilities is the following [13]:

\[
\pi(s_t, a_t) = \frac{\exp[Q(s_t, a)]}{\sum_{a' \in A_t} \exp[Q(s_t, a')]} .
\]

Based on (8), there can be a great difference between the selection probabilities of different actions depending on their values.

### 3.4 Practical issues

In previous section we have described the general design of our routing framework. But for the protocol to be practical, there are additional challenges, which we discuss in detail below.

• **Delayed feedbacks**: The destination is responsible for collecting the throughput measurements and sending them to the source. The simplest approach is to piggyback such information in dedicated end-to-end messages sent by the destination at the end of the batch transmission. In any case, the source’s agent cannot receive the feedback signal (due to end-to-end transmission delays) in time for selecting the \( \eta \) value for the successive batch. In addition, sending a feedback message after each received batch would generate an excessive overhead. To address this issue, in our implementation the destination generates a feedback message only after receiving \( m \) consecutive batches of packets. Furthermore, this allows the agent to apply averaging to the per-batch throughput measurements, thus mitigating fluctuations due to transient conditions. Then, the source uses the same \( \eta \) value for at least \( m + 1 \) consecutive batches. Our results indicate that \( m = 10 \) provides a good tradeoff between responsiveness, stability of throughput estimates and efficiency.

• **Packet losses**: Packets can get lost in the network for several reasons. Generally, unicast routing protocols are totally unaware of these losses that are recovered through mechanisms implemented at different layers of the protocol stack (e.g., using layer-2 or layer-4 retransmissions). On the contrary, most of the opportunistic routing protocols directly retransmit lost packets. Since the \( \eta \) parameter is a measure of the routing efficiency for individual packet transmissions, retransmitted packet should not be included in such computation.

• **Implementation issues**: There are several details to take into account during practical implementation. First of all, we must decide a suitable size for the batch. Note that, differently from other routing schemes that operate on batches of packets, such as ExOR [2], it is not necessary that the source collects a full batch of packets before starting the forwarding process. In our implementation, the batch is a virtual concept used to facilitate the learning process, and to easily allocate each packet to one of the two routing options so as to respect the \( \eta \) fraction. Thus, each packet is forwarded as soon as it reaches the head of the transmission buffer. In our implementation, the batch size is set equal to \((K+1)\) packets.

A second concern regards the measurement of the routing efficiency. In real-world networks the instantaneous throughput may be fluctuating due to a variety of causes (e.g., transient changes of link quality due to fading variations, burstiness of packet arrivals, randomness of channel accesses, etc.). Consequently, it may be difficult to obtain stable estimates of \( \varepsilon = 0 \), as defined in (4). For these reasons, in our implementation we adopt an extended definition of routing efficiency as follows:

\[
\varepsilon_i = \begin{cases} 
1 & \eta_i / \eta > 1 + \beta \\
0 & |\eta_i - \eta| \leq \beta \eta \\
-1 & \text{otherwise}
\end{cases}
\]

In other words, relationship (9) expresses that unicast and opportunistic routing have the same efficiency if the relative difference between \( \eta_i \) and \( \eta \) values is less or equal than \( \beta \). In our implementation, \( \beta = 0.1 \), which is sufficient to absorb small fluctuations of throughput measurements. Finally, since Q-learning is an averaging method that improves its estimates of the average value of action-state pairs as new actions are taken, it is dependent on the initial values of the action-value estimates. In general, it is advantageous to use optimistic initial values to encourage exploration even if greedy actions are selected most of the time. Consequently, in our implementation we start a 2-second probing phase at the beginning of each new connection to get an initial estimate of the efficiency of unicast routing. More precisely, let us assume that in the initial probing phase the traffic is sent using only unicast routing (i.e., \( \eta = 0 \)). In addition, let \( \rho^U \) be the offered load per unit time measured at the source and \( \rho^\prime U \) the unicast throughput measured at the destination at the end of the probing phase. Then, \( \tilde{\eta} = \rho^U \rho^\prime U \) is a rough estimate of the efficiency of unicast transmissions. Thus, a good way
for calculating an initial guess for the values of the action-state function is to assume that states with $\eta > \bar{\eta}$ would not significantly contribute to increase the system reward. More formally, this is equivalent to say that $Q(s, a) = \rho^U$ for all $s = \{\eta, \epsilon\}$ such that $\eta \leq \bar{\eta}$, and $Q(s, a) = 0$ otherwise.

4. PERFORMANCE EVALUATION
A comprehensive simulation study has been conducted to compare the performance of the proposed self-adaptive routing protocol, hereafter simply called Hybrid, against two alternative routing schemes. We first show the gains in terms of throughput improvements in single flow scenarios. Then, we analyze scenarios with multiple flows and unreliable channels.

4.1 Simulation environment
To carry out the following simulation we use the ns-2 network simulator, which implements the full protocol stack for multi-hop wireless networks. We consider wireless mesh networks of 25 nodes, each of which is equipped with one omnidirectional radio antenna. These static nodes are placed randomly in a 500m$\times$500m area. Concerning the physical layer characteristics, to make more realistic simulations we use the Shadowing propagation model instead of the classical TwoRay propagation model, because it permits to describe the received power at a certain distance as a random variable. The following results have been obtained by setting the path loss exponent equal to 2 and the shadowing deviation to 4, which are typical values for outdoor environments [1]. If not otherwise stated the receiving threshold for the network interface is set in such a way to ensure a 95% correct reception rate at the distance of 100m. Concerning the MAC layer, we use the 802.11 DCF scheme with a fixed transmission rate equal to 11 Mbps. Moreover, we have disabled the RTS/CTS access method, since this is the default setting in most wireless networks.

As motivated in Section 1, for the purpose of evaluation we use three routing protocols. First, we use OLSR [7] as a representative of proactive link-state routing protocols, which discovers and then disseminates link state information throughout the mesh network. Individual nodes use this topology information to compute next hops for all destinations in the network using shortest hop forwarding paths and ETX routing metric [8]. Regarding the opportunistic routing option, we use the PacketOPP protocol [5], a lightweight opportunistic routing algorithm able to select at each hop, and at run-time, the candidate forwarders that can maximize the opportunistic throughput gain. Finally, the third protocol is Hybrid, which is a dynamic combination of OLSR and PacketOPP obtained by applying the framework described in Section 3. In the following simulations, to generate the data traffic we use constant bit-rate UDP flows. We do not use TCP-controlled data transfers because traditional TCP would experience unacceptable performance degradations with opportunistic routing due to the frequent out-of-order packet deliveries [2].

4.2 Numerical results
4.2.1 Single flow
First, we evaluate the performance of our solution under a single flow scenario. To this end, Figure 2 shows a box-and-whiskers plot depicting the quartiles of the distribution of flow throughput for OLSR, PacketOPP and Hybrid with different action-selection strategies. The results shown in Figure 2 have been obtained as follows. First, we randomly select a node pair in the network as source and destination of an UDP flow, which sends packets with a payload of 1000 bytes at an offered rate of 1 Mbps. Then, ten statistically independent runs are repeated with this node-pair selection to obtain average values and confidence intervals for the flow throughput. Since throughput performance of individual flows are highly variable, we repeat the same set of simulations with 80 different node pairs to collect a number of samples sufficiently large to obtain good estimates of quartile ranges.

Several important considerations can be derived from the above diagram. First, if we consider the min-max interval traditional unicast routing shows the largest range, and in particular the lowest minimum throughput. By inspecting the traces we discovered that unicast routing achieves very low throughput for source-destination pairs that are separated by many hops. In these cases PacketOPP can take advantage of multiple forwarding nodes and more opportunities to use long links providing throughput gains of a factor of two or more. On the other hand, due to its self-adaptive characteristics, Hybrid protocol is able to select the routing option most suitable for both particularly disadvantaged flows and flows with the highest throughputs. The second observation regards the statistical dispersion of throughput values, i.e., the range between the first quartile and the third quartile, also known as interquartile range. It is important to note that a high variability of the throughput...
obtained by individual node pairs is unavoidable due to the significant differences in the length of shortest routes connecting such pairs, which span from single hop traditional routes to routes that have six hops. However, Figure 2 indicates that OLSR has the largest variability, with many flows having low throughputs. This results in a quite low median throughput. On the other hand, PacketOPP obtains a median throughput 50% higher than OLSR. Despite that, the throughput distribution for PacketOPP is quite concentrated around the median value, which means that only a few flows can achieve high throughputs. On the contrary, Hybrid protocol shows the best performance in terms of interquartile range, which is more shifted towards high throughputs. In addition, Figure 2 shows that Hybrid protocol behaves almost equally well with all the considered policies. However, the pure greedy strategy behaves slightly better than the other considered strategies (i.e., \( \varepsilon \)-greedy methods and softmax method), as it provides the highest median throughput. In the following, all the reported results (except for convergence times) are obtained using the greedy version of our Hybrid scheme.

To better understand the primary reason of throughput gains provided by Hybrid protocol over the other routing strategies, Figure 3 shows the throughput relationship between OLSR and Hybrid with a greedy action-selection policy for the 80 node pairs considered in Figure 2, while Figure 4 shows the relationship between PacketOPP and Hybrid. Figure 3 indicates that there is a large number of node pairs with low unicast throughputs that obtain a significant gain when using Hybrid protocol. Specifically, for half of the node pairs Hybrid outperforms OLSR by more than 50%. Moreover, the node pairs with the highest throughput (mainly pairs with single hop routes) are not affected by the use of Hybrid routing. This means that the overhead introduced by Hybrid (i.e., feedback messages and additional information in packet headers) have a negligible impact. On the contrary, Figure 4 indicates that the throughput gain of Hybrid over PacketOPP occurs mainly for node pairs with mid to high throughput values, while it provides similar performance for node pairs with lower throughputs. To clarify the routing selection behaviour, Figure 5 shows the probability mass function (PMF) of the average \( \eta \) value for the 80 considered node pairs and a greedy action-selection policy. To plot such curve we first compute the average fraction of traffic that was delivered using unicast routing for each individual node pair, and we derive the distribution of such measurement over all the flows. Interestingly we can observe that the PMF of the \( \eta \) parameter shows two principal modes, i.e., two distinct peaks (local maxima). More precisely, most of the flows converge towards \( \eta \) values in the range \([0.25, 0.4]\), which means that these flows use a combination of unicast and opportunistic transmissions, but give preferences to opportunistic routing. Then, there is a second smaller group of flows that converge towards \( \eta \) values close to one, which are mostly node pairs that are separated by a few hops. Interestingly, there is also a non-null probability that a flow uses only opportunistic routing, i.e., \( \eta \leq 0.1 \).

### 4.2.2 Impact of the number of flows

We now turn to consider more scaled scenarios. Basically, we repeat the same simulations discussed in the previous section, but increasing the number of parallel data transfers active in the network. Figure 6 reports average values and 95% confidence intervals for the network capacity (i.e., the sum of the throughputs of individual flows) versus the number of flows. These results are obtained repeating each test with twenty different combinations of selected flows. Note that flow patterns are chosen in such a way that each node can be at most the source/destination of one flow. The plot indicates that Hybrid significantly outperforms OLSR in all the considered scenarios, with throughput gains ranging from 25% to 55%. On the other hand, Hybrid provides smaller improvements with respect to PacketOPP for low numbers of flows (up to 18%), and attains comparable performance for higher numbers of flows. It is important to note that the primary objective of our solution is not to define a new routing protocol, but to show how existing routing paradigms can be adaptively combined to optimize the network performance in the widest range of scenarios. In fact, a specific routing protocol can achieve better performance than another routing protocol in one scenario, but worse...
in another one. Our routing framework is capable of discovering the best routing option for every scenario using a minimal knowledge of the network behaviour and limited overheads.

4.2.3 Impact of shadowing intensity

We investigate the impact of the shadowing intensity on the routing performance. More precisely, we keep constant the path loss exponent (equal to 4) and the shadowing deviation (equal to 0.4), but we vary the receiver sensitivity threshold to change the probability of packet distortion at the distance of 100m. Figure 7(a) shows the average throughput obtained in the case of a single flow, while Figure 7(b) shows the network capacity with ten randomly-selected simultaneous flows. We do not report results for packet-corruption probabilities higher than 0.4 because most of the links are too unreliable to allow OLSR to discover shortest paths between all node pairs. The results confirm that Hybrid efficiency is not affected by the channel unreliability. It is also interesting to observe that Hybrid has similar performance than PacketOPP when the probability of packet corruption is low, while it provides up to 20% throughput gains in more challenged environments. Moreover, unicast routing provides the worst network capacity in all the considered scenarios.

5. CONCLUSIONS

There are many possible application scenarios related to wireless mesh networks. Thus, it is very difficult to design a general routing solution fitting the QoS needs of different applications. Motivated by this, in this paper we have proposed and evaluated a self-adaptive routing framework for wireless mesh networks, which enables the dynamic and on-the-fly combination of multiple routing strategies to maximize routing performance under arbitrary target objectives. The proposed framework relies on efficient machine learning techniques, which allows us to tackle the complexity of the problem optimization without requiring to know how system performance depend on routing primitives and network conditions. Finally, using ns-2 simulations we have shown the superior performance of a hybrid routing protocol obtained through the combination of two legacy schemes.

6. REFERENCES