2009 Eighth IEEE International Symposium on Network Computing and Applications

RL-based Queue Management for QoS Support in Multi-Channel Multi-Radio Mesh Networks

Yu Zhou[1], Mira Yun[1], Timothy Kim[1], Amrinder Arora[2], Hyeong-Ah Choi[1]
[1]Department of Computer Science, The George Washington University, Washington, DC
[2]Department of Computer Science, Bowie State University, Bowie, MD
Email: {yuzhou, mirayun, timothyk, hchoi}@gwmail.gwu.edu, aarora@bowiestate.edu

***Abstract*—In this paper, we consider the important aspect of Quality of Service (QoS) in Wireless Mesh Networks, focusing on packet delays and packet drops. We observe that the options of solely focusing on throughput, and of only depending on the QoS type characterization by the application level protocol is not sufficient. We propose the use of multiple queues to hold the packets based on their QoS requirement (not necessarily corresponding to the QoS application types). We suggest the technique of reinforcement learning (RL) for the important step of assigning packet to one of the queues, and present an implementation using the TD(0) algorithm. We present various simulation results comparing our algorithm to other known algorithms and frameworks. Our results indicate that our RL-based algorithm is significantly better than previously known results in packet drop ratio.**

I. Introduction

Multi-radio multi-channel Wireless Mesh Networks (WMNs) are envisioned as one of the key infrastructures in the next generation of wireless networks. The anticipated role of WMNs has gradually increased from small community/rural/neighborhood networks to also include enterprise-wide networks and other ad hoc wireless backbone networks, such as a disaster recovery network. In order to be successful, these infrastructure networks must meet the QoS requirements from a rich set of applications, including existing applications, mission critical applications, and emerging multimedia applications. A wide variety in the anticipated network topologies also imposes a significant onus on QoS parameters in these networks. A more complete discussion of QoS related challenges in WMNs can be found in [2].

In such multi-radio multi-channel environment, the problem of link and channel scheduling is a critical factor in optimizing the network interferences and throughput. The network must decide which links should transmit at what time, and over which channels. Further, as each node receives packets, it can decide the relative priority that that packet enjoys as compared to other packets waiting at that node.

Many channel assignment strategies have been proposed in the previous research work. [3] provides the Greedy Maximal Scheduling (GMS) generalization for multi-channel network with low-complexity and the same level of efficiency-ratios as single-channel networks. In [4], the author proposes a tree-based 802.11 WMNs architecture with channel assignment and routing algorithms. A distributed channel scheduling algorithm that guarantees the same efficiency ratio as the centralized GMS algorithm in multi-channel wireless networks is proposed in [3]. [5] provides a method in which QoS is supported in WMNs via admission control and routing algorithm at the same time. Further improving the QoS support in lower layer, [6] applied Differentiated Queueing Service (DQS) in WMNs. However, DQS does not address the problem associated with multiple channel environment. In this paper, we propose an algorithm that takes into account differentiated queue management and channel assignment in multi-channel multi-radio environment using Reinforcement Learning [1].

Rest of the paper is organized as follows. In Section II, we present the system model and define the problem. In Section III, we describe our proposed approach, and present the results in Section IV. Our conclusions are presented in Section V.

II. System Model and Problem Statement

In this section, we first describe a network model considered in this paper. Then, we present the problem for queue and channel assignment and the packet scheduling.

A. System Model

We consider a time slotted multi-hop WMN, represented by a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$, where the node set $\mathcal{V}$ denotes the transmission nodes and the edge set $\mathcal{L}$ denotes the links. Each directed edge $(v_i, v_j) \in \mathcal{L}$ represents a link where packets can be transmitted from the transmitter node v_i to the receiver node v_j. If there exist two nodes v_i and v_j whose distance is in the transmission range, two links (v_i, v_j) and (v_j, v_i) will be generated respectively. For a link $l \in \mathcal{L}$ in a transmission, the transmitter and the receiver nodes are denoted by $b(l)$ and $e(l)$, respectively. For a node $v \in \mathcal{V}$, the set $\mathcal{L}(v)$ denotes the set of links attached to node v.

The network has multiple data channels used for data transmission, and each one is on a separate frequency. Let $\mathcal{C}$ denote the set of data channels in the system. It is assumed that each node v is equipped with $\delta(v)$ radios for data transmission such that at any time, v can be involved in up to $\delta(v)$ transmissions as either transmitters or receivers. Each radio can periodically switch from one channel to another according to the channel assignment algorithm. The network also uses an additional control channel to maintain the network topology on a dedicated radio. In this paper, the time is equally divided

978-0-7695-3698-9/09 $25.00 © 2009 IEEE
DOI 10.1109/NCA.2009.46

into slots of unit length, and the overhead of switching between two channels can be negligible. For simplicity, we assume all packets have the same size in our network and $\psi(l,c)$ denotes the transmission rate (packets/per time slot) the link l can provide on channel c, i.e., at each time slot, at most $\psi(l,c)$ packets can be transmitted from the transmitter to the receiver on channel c in link l as long as the link-channel pair (l,c) is not interfered by others. We assume a general interference model, in which for each link-channel pair (l,c), $l \in \mathcal{L}$ and $c \in \mathcal{C}$, there is a set of link-channel pairs $\mathcal{I}_{(l,c)}$ that interfere with (l,c). When (l,c) is in transmission, the link-channel pairs in $\mathcal{I}_{(l,c)}$ can not transmit packets during that timeslot.

There are S data flows in the system and each one is specified by a source node and a destination node. We assume the packets are generated by flow s at the source with a rate λ_s and the traffic of s follows a fixed path during the scheduling period, since the routing table is able to remain steady in a certain period in the reality. We also assume both the transition delay and propagation delay for a packet from one node to another can be ignored, and the queueing delay is the only delay factor we consider in this paper.

B. Problem Statement

We are interested in designing an efficient approach to improve the overall network data throughput, and simultaneously minimize the packet drop ratio.

- **GIVEN:** A graph $\mathcal{G} = (\mathcal{V}, \mathcal{L})$ with interference model $\mathcal{I}$ and arrival process λ
- **TO DO:** Schedule the packets on link channel pairs
- **SUCH THAT:** Total number of packet drops is minimized

III. Proposed Technique

In our system, the WMN provides the basic QoS supports on packet delay. According to the maximum end-to-end delay, each packet is categorized into a corresponding QoS class. In the network, each link $l \in \mathcal{L}$ maintains a set of K queues, $\mathcal{Q}_l = \{q_l^0, q_l^1, q_l^2, ..., q_l^{K-1}\}$, which has decreasing scheduling priorities. We have several principles for the queue design,

- When a packet arrives at a link, it will be inserted at the end of an appropriate queue based on its QoS requirements.
- Once a packet has entered a queue, it can not be relocated to another queue.
- In the same link, the packets in a queue with higher priority are always transmitted before the packets in a queue with lower priority.

If a packet needs to be transmitted earlier, it should be placed in a queue with a higher priority, but conversely, all packets in the queue with a relatively lower priority will have an increased probability to be delayed since that packet has been inserted before them. We need find an appropriate queue for a packet according to how "urgent" this packet is to reach its maximum end-to-end delay. The *urgency* reflects the probability that the packet will be dropped due to exceeding its maximum end-to-end delay, and it is able to be represented as a utility by a marginal utility function.

Since the resource of channels and radios are limited, some links can not be simultaneously scheduled due to the interference. In the next section, we will present our approach of the queue and channel assignment and packet scheduling using reinforcement learning.

A. Reinforcement Learning

The reinforcement learning is a methodology aiming to provide a solution of minimizing (maximizing) the long-term cost (reward) through iterations of choosing appropriate actions. The learner learns the behavior by trial-and-error interactions and finally figures out a correct or optimal answers during the iterations in a dynamic system.

A standard RL model consists of

- A set of states $\mathcal{S}$
- A set of actions $\mathcal{A}$
- An expected immediate cost R under the cost function $\mathcal{S} \times \mathcal{A} \rightarrow \mathcal{R}$. Given the current state $s \in \mathcal{S}$ and the action taken $a \in \mathcal{A}$ (determined by the policy) at iteration n, $R_{ss'}^a$ amount of cost is generated if the next state is transited to s', i.e., $R_{ss'}^a = E\{r_n | s_n = s, a_n = a, s_{n+1} = s'\}$, where $r_n = r(s_n, a)$ represents the instant scalar cost generated at iteration n.

The *value function* of a state s under a policy π, denoted $V^\pi(s)$, is defined as the expected cost of the learner starting from state s and following policy π thereafter, i.e., $V^\pi(s) = E\{\sum_{k=0}^{\infty} \gamma^k r_k | s_0 = s\}$, where $\gamma (0 \le \gamma \le 1)$ is called the *discount factor*. In the RL, the ideal is to find out an optimal policy π^* which achieves a minimized long-term cost. The *optimal value function* of state s, denoted $V^*(s)$, is defined as $V^*(s) \equiv V^{\pi^*}(s) = \min_\pi V^\pi(s)$.

Temporal Difference (TD) [1] is a general approach of RL to update $V(s)$ in a bootstrapping process. The simplest form TD(0) is a one-step-ahead prediction, and it uses the following equation to update $V(s)$,

$$V(s_n) = (1 - \alpha_n) V(s_n) + \alpha_n \big[r_n + \gamma V(s_{n+1}) \big], \qquad (1)$$

where α_n is the *learning rate*. Given bounded costs $|r_n|$ and the learning rates $0 \le \alpha_n < 1$, the value function $V(s)$ converges to $V^*(s)$ with probability 1 if $\sum_{n=0}^{\infty} \alpha_n = \infty$ and $\sum_{n=0}^{\infty} [\alpha_n]^2 < \infty$ [7].

B. States, Actions, Costs

The WMN model we proposed can be considered as a discrete event system. The events of packet generation, transmission in the WMNs can be modeled as stochastic variables with appropriate probability distributions. In this section we elaborate how to define appropriate states, actions, costs respectively in our WMN model.

1) States: In WMNs, each packet can be considered as a discrete object. In the actual WMN environment, the situation is often observed that multiple packets with the same source and destination arrive at one node successively and usually these packets have similar QoS requirements. Consequently, these successively arriving packets have large probability to be assigned to the same queue. We say link l is in the state of preferring queue q if the current packet is assigned to q. Therefore, the state s_n at iteration n is defined as $s_n = (l,q)_n$, where $l \in \mathcal{L}, q \in \mathcal{Q}_l$, which is a combination of the link and the queue.

2) Actions: For a link l, the state of l might change whenever a packet coming in. The queue choice, say q_l, of the current packet is defined as an action a, such that $a = q_l$, where $q_l \in \mathcal{Q}_l$. There are K possible actions, i.e., choices, for a packet when it is choosing one queue out of K. The link state does not change if a packet is leaving from the queue.

3) Costs: The cost $r(s,a)$ indicates the immediate cost that the system generates at interation n when action a is taken at state s. In the network mode we proposed, that is equivalent to the cost the system generates when packet p is assigned to queue $q \in \mathcal{Q}_l$. First, we quantify the "urgency" of packet p using a marginal utility function $U(p)$. Suppose there are H hops for p on its routing path from the source to the destination, and the maximum end-to-end delay of p is D. Therefore, on the average, the maximum tolerable queueing delay at each node is $\frac{D}{H}$. p is created on time t_0 and it is in the dth node on its path at the current time t. $d = 0$ means the source node on the path.

$$U(p) = \begin{cases} \epsilon & \text{if } t - t_0 \le \frac{dD}{H} \\ \left(\frac{t-t_0-\frac{dD}{H}}{\frac{D}{H}}\right)^{(H-h)^{\phi}} & \text{if } \frac{dD}{H} < t - t_0 < \frac{(d+1)D}{H} \\ 1 & \text{if } t - t_0 \ge \frac{(d+1)D}{H} \end{cases} \tag{2}$$

where ϵ denotes an extremely small positive value close to 0 such that any action is guaranteed to generate a valuable cost. ϕ denotes an urgency factor with $\phi > 0$ and we have $\phi = 3$ in our simulation. We can see that $U(p)$ is a function with range $(0,1]$. The larger value of $U(p)$ indicates that p is more urgent to be scheduled and it should be more likely to be assigned to a queue with a higher QoS priority.

Suppose packet p resides in queue $q_i^l \in \mathcal{Q}_l$, and p is the jth $(j \ge 0)$ packet in queue q_i^l, where $(0 \le i \le K-1)$. Let $N(q)$ denotes the number of packets in q, and $N(l)$ denotes the number of packets in l, $N(l) = \sum_{q \in \mathcal{Q}_l} N(q)$. Let $L(p)$ denote the absolute location of p in l and it can be represented as $L(p) = j + \sum_{k=0}^{i-1} N(q_k)$. The cost $r(s,a)$ of link l is defined as $r(s,a) = \frac{\sum_p (U(p) \cdot L(p))}{N(l)}$, which reflects the average weighted urgency of the packets in the link. $L(p)$ can be considered as a weight on the urgency $U(p)$ since the larger $L(p)$ may lead to a later scheduling on p which makes p more urgent.

C. Algorithm Design

We now describe the design of our algorithm using TD(0).

1) Queue Assignment: When a packet p arrives at one node, it has to be assigned to one of the K queues. We take a greedy action with respect to the cost $r(s_n,a)$ and value function $V(s_{n+1})$, a "better" policy π of choosing queue q^π is updated continually by the equation,

$$q^\pi \equiv a^\pi = \operatorname*{argmin}_{a \in \mathcal{A}} E\{r(s_n,a) + \gamma V(s_{n+1})\}, \tag{3}$$

From action a^π, we keep the average weighted urgency of each link as small as possible, this is reasonable because we could figure out those links which are really in urgency then they can be scheduled in the higher priority. From (2), we know that $U(p)$ is in the range of 0 and 1, and $L(p)$ is also bounded by the summation of queue length, such that cost $r(s_n,a)$ is bounded too. In the simulation, we define $\alpha_n = \frac{1}{n}$. Therefore, from the previous description in III-A, we know the value function $V^\pi(s)$ will always converge to $V^*(s)$ after sufficient iterations following equation (1).

2) RL-based Distributed LubyMIS Channel Assignment: The channel assignment algorithm is applied every ρ time slots. In the algorithm, we use the Luby Maximal Independent Set (LubyMIS) algorithm [8] for each channel c. The algorithm consists of three rounds. In the first round, each link updates its link-channel pair weight $w(l,c,t)$ and send it to interference neighbors through the control channel, where $w(l,c,t)$ is defined as $w(l,c,t) = V(s) \cdot N(l) \cdot \psi(l,c)$, and s is the current state of link l. By the end of the first round, links with highest weight are marked as the winner. In the second round, each winner notify their interference neighbors the fact that they have won. Thus at the end of second round, the interference neighbors knows that they are the losers. In the third round, each loser notifies its neighbors. Then all the winners, the losers, and the loser's neighbors remove the appropriate link-channel pairs from the network. After the third round, the algorithm repeats from the first round to find the winners, the losers, and the loser's neighbors in remaining nodes and links. This process is repeated until no more link-channel pairs are left. The algorithm implementation is showed in Algorithm 1.

IV. Simulation Results

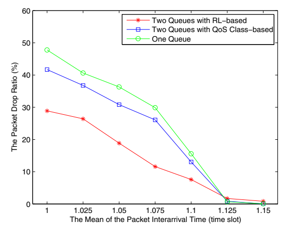
Our simulation runs on a 5×5 grid topology where each node could potentially communicate with up to four neighbors. We create 10 data flows by randomly picking 10 source-destination pairs in the network, and each flow has a predefined fix routing path. The network supports 2 QoS classes which have maximum end-to-end delay of 50 and 100 time slots respectively. The number of queues K is also initiated as 2 in the simulation. Each data flow is randomly assigned with one QoS class. All packets in the same data flow have the same QoS requirements. The number of radios used for data transmission on each node varies from 1 to 3 . Each radio has 12 non-overlapping data channels. For simplicity, all channels have the fixed rate with $\psi = 5$ packets per time slot. In the simulation, the packets are generated at each source node following an exponential distribution. We respectively measure the packet drop ratio of the network with different mean values of the packet interarrival time varying from 1 to 1.15 time slots. Every 5 time slots the channel assignment algorithm is called and the overall simulation time is 10000 time slots. The discount factor γ is 0.8 and the urgency factor ϕ is 3 in the simulation.

Algorithm 1 RL-based Distributed LubyMIS Channel Assignment

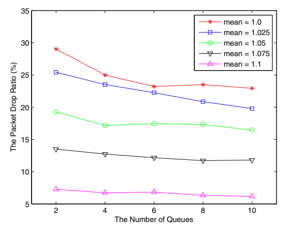
```
$t \leftarrow$ current time slot;
Let $\mathcal{Z} = \{(l,c) | l \in \mathcal{L}, c \in \mathcal{C}\}$;
Let $\mathcal{F} = \emptyset$;
Initialize $\beta(v) \leftarrow \delta(v)$ for all nodes $v$;
Define $w(l,c,t) = V(s)N(l)\psi(l,c)$;
for $c \in \mathcal{C}$ do
    Let $\mathcal{Z}' = \{(l,c) | (l,c) \in \mathcal{Z}\}$;
    while $size(\mathcal{Z}') > 0$ do
        use LubyMIS to find $(l,c) \in \mathcal{Z}'$ which has the largest weight $w(l,c,t)$ in $(l,c) \cup (\mathcal{I}_{(l,c)} \cap \mathcal{Z}')$;
        $(l,c) \leftarrow$ winner;
        $\beta(b(l)) \leftarrow \beta(b(l)) - 1$;
        $\beta(e(l)) \leftarrow \beta(e(l)) - 1$;
        remove $(l,c)$ from $\mathcal{Z}'$ and $\mathcal{Z}$;
        add $(l,c)$ into $\mathcal{F}$;
        for all $(l',c) \in (\mathcal{I}_{(l,c)} \cap \mathcal{Z}')$ do
            remove $(l',c)$ from $\mathcal{Z}'$ and $\mathcal{Z}$;
        end for
        if $\beta(b(l)) = 0$ then
            for $k \in \mathcal{L}(b(l))$ do
                Remove $(k,c')$ from $\mathcal{Z}$ for all channels $c'$;
            end for
        end if
        if $\beta(e(l)) = 0$ then
            for $k \in \mathcal{L}(e(l))$ do
                Remove $(k,c')$ from $\mathcal{Z}$ for all channels $c'$;
            end for
        end if
    end while
end for
return $\mathcal{F}$;
```

In the first scenario, we evaluate the performance of the RL-based LubyMIS algorithm we proposed by comparing it against another two algorithms. In the first algorithm, only one queue is available in each link. All packets are sequentially inserted into the queue according to their incoming order and scheduled in the way of FIFO. The second algorithm provides crude QoS considerations. Each QoS class has its corresponding queue in the link. A queue is simply assigned to each incoming packet according to the QoS class. The channel assignment in these two algorithms is similar as the algorithm we proposed, except that $N(l)\psi(l,c)$ is used instead of $w(l,c,t)$. The results are shown in Figure 1(a). We notice the RL-based LubyMIS algorithm can provide a better QoS performance on packet drop ratio compared with the other two algorithms. With the increasing of the packet interarrival time, the packet drop ratio decreases almost linearly in our algorithm. When the mean value of the packet interarrival time reaches 1.125, the traffic load in the network is low enough such that it can be well supported with a low packet drop ratio no matter what algorithm is used.

In the second scenario, we measure the impacts of the number of queues K on the packet drop ratio using the RL-based LubyMIS algorithm. The simulation runs with different mean values of the packet interarrival time from 1.0 to 1.1 using our algorithm. Figure 1(b) presents the simulation results. Generally, the network with the larger number of queue is able to provide better performance since each packet can choose a queue more accurately based on its weighted delay urgency, that is also to say, each state of RL can be represented more accurately such that the action made on the state is more likely close to the optimal. But the system can not keep achieving better performance with the increment of K. Once K exceeds some threshold, the system performance keeps stable.



(a) The packet drop ratio of different algorithms



(b) The packet drop ratio with different value K

Fig. 1. Simulation results

V. Conclusion

In this paper, we proposed a Reinforcement Learning based queue management scheme that supports QoS in multi-radio multi-channel mesh networks. Simulation results show that our proposed multiple queues scheme significantly performs better than other existing queue management scheme, namely using a single queue or using multiple queues classified based on application layer QoS types. It should be also noted that the performance of our algorithm improves when the number of queues increases but only up to a certain threshold point. For future work, we plan to consider the convergence speed of value function in the RL design. How to control the convergence speed using an appropriate learning rate in our model is still a problem which needs to be considered.

References

[1] A. G. Barto and R. S. Sutton. *Reinforcement Learning: An Introduction.* MIT Press, 1988.

[2] P. Mogre, M. Hollick, and R. Steinmetz. *QoS in Wireless Mesh Networks: Challenges, Pitfalls, and Roadmap to its Realization*. In NOSSDAV 2007, June 2007.

[3] X. Lin and S. Rasool. *A Distributed Joint Channel-Assignment, Scheduling and Routing Algorithm for Multi-Channel Ad-hoc Wireless Networks*. In INFOCOM 2007, pages 1118-1126, May 2007.

[4] A. Raniwala and T. Chiueh. *Architecture and algorithms for an IEEE 802.11-based multi-channel wireless mesh network*. In INFOCOM 2005, volume 3, pages 2223-2234, March 2005.

[5] X. Cheng, P. Mohapatra, S.-J. Lee, and S. Banerjee. *MARIA: Interference-aware admission control and QoS routing in wireless mesh networks*. In ICC'08, pages 2865-2870, May 2008.

[6] X. Teng, S. Jiang, G. Wei, and G. Liu. *A cross-layer implementation of differentiated queueing service (dqs) for wireless mesh networks*. Vehicular Technology Conference, 2008. VTC Spring 2008. IEEE, pages 2233-2237, May 2008.

[7] J. Tsitsiklis and B. Van Roy. *An analysis of temporaldifference learning with function approximation*. Automatic Control, IEEE Transactions on, 42(5):674-690, May 1997.

[8] M Luby, *A simple parallel algorithm for the maximal independent set problem*, Proceedings of the seventeenth annual ACM symposium on Theory of computing, New York, NY, USA, 1985, pp. 1-10, ACM.